Interfaces and Interoperability After *Google v. Oracle*[†]

Mark A. Lemley[*] & Pamela Samuelson[**]

Introduction

In its landmark decision, *Google LLC v. Oracle America, Inc.*,[1] the U.S. Supreme Court declined to address what had long been Google's main argument against Oracle's decade-old lawsuit charging it with copyright infringement for reusing parts of the Java Application Programming Interface (API) in its Android smartphone software.[2] The Court "assume[d], but purely for argument's sake, that the entire Sun Java API falls within the definition of that which can be copyrighted."[3] It nonetheless found Google's reimplementation of those API elements legal under copyright's fair use doctrine because it enabled Java programmers to make use of their investments in learning the Java API declarations at issue to develop new application programs that could interoperate with the Android platform.[4]

† © 2021 Mark A. Lemley & Pamela Samuelson.

* William H. Neukom Professor, Stanford Law School; Partner, Durie Tangri LLP.

** Richard M. Sherman Distinguished Professor of Law, Berkeley Law School. Thanks to Clark Asay, Jonathan Band, Jim Bessen, Joshua Bloch, Oren Bracha, Michael Carrier, Courtney Cox, Jim Dabney, Eric Goldman, Paul Goldstein, James Grimmelmann, Rose Hagan, David Hayes, Jonathan Masur, Sancho McCann, Peter Menell, Aaron Perzanowski, Jerry Reichman, Xiyin Tang, and Rebecca Tushnet for comments on a prior draft.

1. 141 S. Ct. 1183 (2021).

2. *Id.* at 1197. Justice Breyer authored the Court's opinion for himself and five other Justices. Justices Thomas and Alito dissented. *Id.* at 1210. Early in the litigation, Google moved for summary judgment on its defense that the interface elements at issue were uncopyrightable. The trial judge, William Alsup, denied this motion. *See* Oracle Am., Inc. v. Google Inc. (*Oracle I*), 810 F. Supp. 2d 1002, 1005, 1013 (N.D. Cal. 2011) (denying Google's summary judgment theories with one exception for "the names of various items appearing in the disputed API package specifications"). After a trial on the merits, Judge Alsup made findings of fact and rendered conclusions of law in support of Google's uncopyrightability defense. Oracle Am., Inc. v. Google Inc. (*Oracle II*), 872 F. Supp. 2d 974 (N.D. Cal. 2012). Oracle successfully appealed that decision. Oracle Am., Inc. v. Google Inc. (*Oracle III*), 750 F.3d 1339 (Fed. Cir. 2014). The Supreme Court denied Google's first petition for certiorari on the copyrightability issue. Google Inc. v. Oracle Am., Inc., 576 U.S. 1071 (2015). Upon remand, a jury rendered a verdict in favor of Google's fair use defense. Judge Alsup denied Oracle's Rule 50 motion for a judgment as a matter of law. Oracle Am., Inc. v. Google Inc. (*Oracle IV*), No. C 10-03561 WHA, 2016 WL 5393938, at *1 (N.D. Cal. Sept. 27, 2016). Oracle successfully appealed that decision and persuaded the Federal Circuit that no reasonable jury could have found fair use. Oracle Am., Inc. v. Google LLC (*Oracle V*), 886 F.3d 1179 (Fed. Cir. 2018). The Supreme Court's reversal of *Oracle V* did not, however, vacate *Oracle III*.

3. *Google*, 141 S. Ct. at 1197. The Court's explanation for ducking the copyrightability question was that "[g]iven the rapidly changing technological, economic, and business-related circumstances, we believe we should not answer more than is necessary to resolve the parties' dispute." *Id.*

4. *Id.* at 1208–09.

The Court's decision to sidestep the copyrightability issue is notable because Google, and all but three of the twenty-seven amicus curiae briefs filed in support of its appeal, urged the Court to hold that the Java API declarations that Google reimplemented in Android were unprotectable by copyright law under copyright law's merger doctrine and/or the statutory exclusion of systems and methods of operation from the scope of copyright protection.[5]

The Court's sweeping fair use ruling is an important victory for software developers and for an open internet. But the decision not to address the larger question—whether interfaces are copyright-protectable at all—will produce uncertainty. Because the Court sidestepped the copyrightability issue in *Google*, Federal Circuit judges may well decide (wrongly) that the Court implicitly accepted its earlier holding that software interfaces are copyrightable.[6] And while, as we show, the regional circuits are virtually unanimous in refusing to allow the copyrighting of interfaces, those who want to claim copyrights in their interfaces can assure themselves that the Federal Circuit will hear any appeal of copyrightability rulings by a simple trick of forum shopping, namely, adding a patent claim to their complaints. A case in point now pending before the Federal Circuit is *SAS Institute, Inc. v. World Programming Limited*,[7] in which a district court ruled that WPL's reimplementation of SAS interface elements did not infringe SAS's copyright because SAS did not prove that those elements were copyrightable.[8] SAS's appeal relies heavily on the reasoning of the Federal Circuit's 2014 *Oracle III*[9] decision.[10]

5. Brief for the Petitioner at 17–19, *Google*, 141 S. Ct. 1183 (No. 18-956). Only two of the twenty-seven Google-side amicus briefs focused solely on the merits of Google's fair use defense—one by Microsoft and one by twenty-five copyright scholars. A third Google-side amicus brief focused on whether fair use should be decided by judges or juries.

6. *See Oracle V*, 886 F.3d 1179 (Nos. 2017-1118, 2017-1202) (per curiam) (non-precedential order), http://www.cafc.uscourts.gov/sites/default/files/opinions-orders/17-1118.ORDER.5-14-2021_1777843.pdf [https://perma.cc/Y2JE-3RVY] (vacating the 2018 fair use ruling in light of *Google* and remanding case to district court).

7. 496 F. Supp. 3d 1019 (E.D. Tex. 2020), *appeal docketed*, No. 2021-1542 (Fed. Cir. Jan. 13, 2021) (Westlaw).

8. *Id.* at 1021. In a nearly identical case between the same parties the Court of Justice of the European Union held that the functional behavior of SAS's program and the SAS Language were not protectable by copyright law. Case C-406/10, SAS Inst. Inc. v. World Programming Ltd., ECLI: EU:C:2012:259, ¶ 46 (May 2, 2012). A North Carolina district court also ruled against SAS's copyright claim against WPL for emulating the functionality of the SAS program, SAS Inst. Inc. v. World Programming Ltd., 64 F. Supp. 3d 755, 783 (E.D.N.C. 2014), which the Fourth Circuit decided was moot after it affirmed relief granted on a different claim, SAS Inst., Inc. v. World Programming Ltd., 874 F.3d 370, 375 (4th Cir. 2017), *cert. denied*, 141 S. Ct. 1053 (2021).

9. 750 F.3d 1339 (Fed. Cir. 2014).

10. *See* Brief for Plaintiff-Appellant, SAS Inst. Inc. v. World Programming Ltd., No. 2021-1542 (Fed. Cir. May 14, 2021) (citing *Oracle III* repeatedly); *see also id.* at 35 n.2 (claiming that the Supreme Court had not reversed the 2014 copyrightability ruling).

In this article, we argue that defendants in future software copyright cases should not shy away from challenging the copyrightability of program interfaces.[11] The Federal Circuit decision in *Oracle III*, which wrongly extended copyright protection to APIs, is contrary to a consensus among multiple appellate precedents concluding that copyright does not protect elements of software programs, such as APIs, that enable interoperability. Those cases remain good law, and the Federal Circuit is supposed to follow them.

Refusing to protect APIs is also good policy. Indeed, we argue that decisions not to give the developers of APIs copyright control over reimplementation three decades ago was central to the development of the interoperable ecosystem and the open internet we enjoy today. Ensuring interoperability, in part by denying copyright protection to those who want to close interfaces, may be the key to restoring an open internet that is today threatened by dominant players with strong incentives to close their networks.

In Part I, we discuss the consensus that emerged in the 1990s and early 2000s among the regional circuits that copyright protection does not extend to interfaces that facilitate the development of interoperable computer programs.[12] Courts have invoked four principal copyright doctrines in support of their holdings that the scope of copyright in computer programs does not extend to such interfaces. One is the *scenes a faire* doctrine, which excludes from copyright protection elements of copyrighted works that are commonplace or constrained by external factors, such as the need to be compatible with other software.[13] A second is the merger doctrine, under which original elements of protected works that can, as a practical matter, be expressed in relatively few ways are said to be "merged" with the idea, fact, or function, and are hence unprotectable by copyright law.[14] A third focuses on the statutory exclusion of procedures, processes, systems, and methods of

11. We do not consider a challenge to the copyrightability of specific elements of computer programs, such as interfaces, to be a "defense" to a claim of infringement. Defendants may need to challenge copyrightability and explain why they think the elements are not protectable by copyright law. However, they do not bear the burden of persuasion on the issue because challenges to copyrightability of specific elements goes to whether the plaintiff has made out a prima facie case of infringement. The burden is always on plaintiffs to prove that what defendants appropriated from their work is protectable expression. *See, e.g.*, PAUL GOLDSTEIN, GOLDSTEIN ON COPYRIGHT § 16.3.2 (3d ed. Supp. 2021–2022) (explaining the burdens of proof for copyright infringement).

12. We use the terms compatibility and interoperability interchangeably in this Article.

13. *See, e.g.*, Restatement of Copyright § 12(d) (Am. L. Inst., Tentative Draft No. 2, 2021) (noting that copyright does not extend to elements of expression that constitute *scenes a faire*).

14. *See, e.g.*, Pamela Samuelson, *Reconceptualizing Copyright's Merger Doctrine*, 63 J. COPYRIGHT SOC'Y USA 417 (2016) (discussing merger doctrine).

operation embodied in copyrighted works.[15] A fourth is the fair use doctrine, which considers the purpose of the challenged use, the nature of the copyrighted work, the amount and substantiality of the taking, and the consequential harms to markets for the protected works arising from the challenged use.[16] While courts sometimes adopted different doctrinal paths, they came to the same conclusion for well over two decades: that program interfaces needed to achieve compatibility are unprotectable by copyright law. Only the Federal Circuit's decision in *Oracle III* deviated from this consensus.

In Part II, we discuss the Federal Circuit's decision in *Oracle III*, which departed from these well-established precedents. The Supreme Court's *Google* decision took a broad view of fair use as a limit on computer program interfaces, and there are hints in the Court's opinion that Google's copyrightability challenge had some merit. Nonetheless, we regard *Google* as a lost opportunity to recognize the important role that § 102(b) exclusions have played in previous software interface cases. Part II also suggests that the Supreme Court may have viewed the *Google* case as distinguishable from earlier interoperability precedents. Most programs developed for the Android platform were not fully interoperable with other Java platforms. The earlier interface-uncopyrightability cases, by contrast, generally sought to achieve complete compatibility. This may help to explain why the Court regarded fair use as a sounder basis for its ruling in Google's favor.

Finally, in Part III we explain why denying copyright protection to interfaces is good policy. Enabling compatibility among programs has become an even more urgent imperative in today's digital ecosystem than in the 1990s, when courts first directly addressed the unprotectability of interfaces. The ability of second comers to reimplement interfaces has made software development faster, lowered barriers to entry, promoted competition and ongoing innovation, and benefited consumers in innumerable ways. The *Google* decision has preserved fair use as a defense to claims of infringement for reimplementing interfaces, but we argue that fair use is an incomplete solution.[17] Courts should reaffirm their longstanding commitment to

15. *See, e.g.*, Pamela Samuelson, *Why Copyright Excludes Systems and Processes from the Scope of Its Protection*, 85 TEXAS L. REV. 1921 (2007) (discussing the statutory exclusion of systems and processes from copyright protection).

16. 17 U.S.C. § 107. For a discussion of this doctrine's application to computer program interfaces, see, e.g., Pamela Samuelson & Clark D. Asay, *Saving Software's Fair Use Future*, 31 HARV. J.L. & TECH. 535 (2018).

17. In our view, fair use is a defense to charges of copyright infringement but not an affirmative defense. That is, defendants must assert fair use, but fair use should not be understood as an affirmative defense in the sense that the defendants bear the ultimate burden of persuasion on it for reasons set forth in Lydia Pallas Loren, *Fair Use: An Affirmative Defense?*, 90 WASH. L. REV. 685 (2015). Unfortunately, courts have opined otherwise. *See, e.g.*, Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569, 590 (1994) (describing fair use as an affirmative defense).

withholding copyright from APIs that enable compatibility. Doing so will clear the path for an open and competitive internet, something that is currently under siege by dominant incumbents with walled gardens.[18]

I. Numerous Courts Have Held That Software Interfaces Necessary for Achieving Compatibility Are Uncopyrightable.

APIs set forth the rules and procedures by which programs are able to exchange information with other programs (or with hardware) so the programs can interoperate and collaboratively carry out specific programmed tasks.[19] A long-standing norm in the computing field has been that everyone should be obliged to develop their own implementation of an interface, but everyone should be free to reimplement interfaces in their own independently written code.[20]

18. *See, e.g.*, Dan Hunter, *Walled Gardens*, 62 WASH. & LEE L. REV. 607, 611 (2005) (advocating for an open-access publishing model in legal scholarship to break through the walled gardens of commercial databases); Lina M. Khan, *The Separation of Platforms and Commerce*, 119 COLUM. L. REV. 973, 1097 (2019) (explaining how digital platforms have an economic incentive to "keep users within their walled gardens" in order to collect as much user data as possible); Salil K. Mehra, *Paradise Is a Walled Garden? Trust, Antitrust, and User Dynamism*, 18 GEO. MASON L. REV. 889, 889–90 (2011) (examining the perception of walled gardens as proprietary and sterile); Greg Lastowka, *Walled Gardens and the Stationers' Company 2.0*, IDP: REVISTA D'INTERNET, DRET I POLÍTICA, Nov. 2012, at 41, 47 (Spain), https://ssrn.com/abstract=2204465 [https://perma.cc/65JU-DRKH] (discussing the problems caused by intermediary tech platforms creating walled gardens and using copyright to remove power from creators); Rob Frieden, The Internet of Platforms and Two-Sided Markets: Legal and Regulatory Implications for Competition and Consumers (Oct. 13, 2017) (unpublished manuscript), https://privpapers.ssrn .com/sol3/papers.cfm?abstract_id=3051766 [https://perma.cc/Z63F-7S3E] (discussing platform monopolies and noting failures in governmental regulation of internet platforms).

19. *Oracle I* explains that:

> Conceptually, an API is what allows software programs to communicate with one another. It is a set of definitions governing how the services of a particular program can be called upon, including what types of input the program must be given and what kind of output will be returned. APIs make it possible for programs (and programmers) to use the services of a given program without knowing *how* the service is performed. APIs also insulate programs from one another, making it possible to change the way a given program performs a service without disrupting other programs that use the service.

810 F. Supp. 2d 1002, 1005 (N.D. Cal. 2011).

Although the Supreme Court's *Google* decision characterized the Java API elements at issue as a "user interface" (because programmers use them to interact with machines), Google LLC v. Oracle Am., Inc., 141 S. Ct. 1183, 1201 (2021), this Article avoids use of that term as it encompasses some elements of programs, such as videogame graphics, that do not implicate the interoperability concerns on which this Article focuses.

20. *See, e.g.*, Alfred Z. Spector, *Software, Interface, and Implementation*, 30 JURIMETRICS J. 79, 88 (1989) (providing the arguments in favor of this norm from a computer scientist's perspective). The Court's *Google* decision observed that the jury heard evidence that "shared interfaces are necessary for different programs to speak to each other" and that "the reuse of APIs is common in industry." *Google*, 141 S. Ct. at 1203–04.

In this Part, we show how software developers first enabled, but later restricted, compatibility by controlling access to APIs and claiming copyright protection in them. By the mid-1990s, courts had reached a consensus that those claims were unpersuasive and that copyright protection did not allow a copyright owner to prevent others from developing similar, compatible programs.

A. *A Pre-History of Interoperability in the Software Industry and Copyright Protections*

In the early days of computing, companies attained commercial success by selling computer hardware, not computer software.[21] Manufacturers of computers tended to bundle computer programs with their platforms to make them more attractive to customers.[22] They had ample incentives to make interfaces for their operating systems (OS) available to their customers and to third parties to encourage them to develop software that would run on their platforms. All programs were tailored to run on the one particular hardware and OS environment for which they were developed; programs had to be rewritten to run on other platforms. There was no such thing as cross-platform or cross-application compatibility. Running a program on a different system meant rewriting that program to work with the new system. Niche service businesses emerged to do software maintenance, system integration, and customization.[23]

Beginning in the mid-1960s and into the late-1970s, questions arose about whether copyrights, patents, or both were or should be available to computer programs. In 1965 the U.S. Copyright Office decided to accept registration of the full source code listings of computer programs under its "rule of doubt."[24] The next year a Presidential Commission recommended

21. *See, e.g.*, JONATHAN BAND & MASANOBU KATOH, INTERFACES ON TRIAL: INTELLECTUAL PROPERTY AND INTEROPERABILITY IN THE GLOBAL SOFTWARE INDUSTRY 18–25 (1995) (describing the computer market industry in the 1950s and 1960s).

22. *See, e.g.*, MARTIN CAMPBELL-KELLY, FROM AIRLINE RESERVATIONS TO SONIC THE HEDGEHOG: A HISTORY OF THE SOFTWARE INDUSTRY 6 (2003) (explaining that companies used to provide computer programs for free to purchasers of their hardware).

23. *See, e.g.*, Stephen Breyer, *The Uneasy Case for Copyright: A Study of Copyright in Books, Photocopies, and Computer Programs*, 84 HARV. L. REV. 281, 344–47 (1970) (surveying software trends that created opportunities for smaller independent companies).

24. *See* COPYRIGHT OFFICE CIRCULAR 31D (Jan. 1965), *reprinted in* Duncan M. Davidson, *Protecting Computer Software: A Comprehensive Analysis*, 1983 ARIZ. ST. L.J. 611, 652 n.72 (1983) ("[I]n accordance with the policy of resolving doubtful issues in favor of registration whenever possible, the Copyright Office will consider registration."); *see also* Davidson, *supra* at 739 (defining "rule of doubt"). The Office expressed doubts about whether machine-executable forms of programs could be copyrighted on account of their functionality. COPYRIGHT OFFICE CIRCULAR 31D, *supra*.

against patent protection for computer program innovations, in part because it perceived copyright protection to be available for software.[25]

Because some meddlesome new technology issues, such as whether copyright was a suitable form of legal protection for programs, were holding up enactment of what would become the Copyright Act of 1976, Congress decided to create a National Commission on New Technological Uses of Copyrighted Works (CONTU) to address these issues and make recommendations about whether Congress should amend the Act to implement the recommendations.[26]

In its 1978 Report to Congress, CONTU endorsed copyright protection for computer programs,[27] albeit over dissents expressing concern about the use of copyright law to protect highly functional works.[28] CONTU recommended a few changes to the 1976 Act to tailor it for programs, which Congress enacted in 1980.[29] CONTU did not address whether interfaces or

25. PRESIDENT'S COMM'N ON THE PATENT SYS., "TO PROMOTE THE PROGRESS OF . . . USEFUL ARTS": REPORT OF THE PRESIDENT'S COMMISSION ON THE PATENT SYSTEM 13 (1966). The Commission states:

> Uncertainty now exists as to whether the statute permits a valid patent to be granted on programs. Direct attempts to patent programs have been rejected on the ground of nonstatutory subject matter. Indirect attempts to obtain patents and avoid the rejection, by drafting claims as a process, or a machine or components thereof programmed in a given manner, rather than as a program itself, have confused the issue further and should not be permitted.
>
> . . . .
>
> It is noted that the creation of programs has undergone substantial and satisfactory growth in the absence of patent protection and that copyright protection for programs is presently available.

Id.

26. CONTU was established by Pub. L. No. 93-573, 88 Stat. 1873 (1974). *See* NAT'L COMM'N ON NEW TECH. USES OF COPYRIGHTED WORKS, FINAL REPORT 34 (1978) [hereinafter CONTU REPORT] (discussing the potential difficulties of extending copyright to computer programs). None of the CONTU Commissioners had expertise in computer or software technologies. *Id.* at App. C. Most of the Commissioners were more concerned about the photocopying issues that Congress wanted CONTU to address than about the computer-related issues. *See id.* at 47–78 (discussing photocopying issue in considerable detail).

27. CONTU REPORT, *supra* note 26, at 15–16 (asserting that computer programs were already copyrightable under the 1976 Act). One of us has criticized the CONTU Report for its misstatements about computer programs, about the unsuitability of copyright law as a form of software protection, and about whether Congress had already decided to protect machine-readable forms of programs. *See* Pamela Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form*, 1984 DUKE L.J. 663 (1984) (recommending instead "a new intellectual property law specifically for machine-readable computer programs").

28. CONTU REPORT, *supra* note 26, at 27–37 (dissent of John Hersey) (arguing that programs were too functional to be copyright subject matter and failed to communicate the works' expression to humans as other copyrighted works do); *id.* at 37–38 (dissent of Rhoda H. Karpatkin).

29. *Id.* at 12–14 (recommendations of majority). *See* Pub. L. No. 96-517, § 10, 94 Stat. 3015, 3028 (1980) (codified as amended at 17 U.S.C. §§ 101 (defining "computer program"), 117 (providing limitation to enable copying incidental to use, backup copying, necessary adaptations, and resales of one's copy)).

any other specific aspects of computer programs were or should be protected by copyright law. Rather, it assumed that computer programs qualified as literary works and that courts would be able to apply traditional copyright doctrines when software developers charged one another with infringement.[30]

In the 1980s, the computer and software industries experienced significant growth. In this period, it became common, although not ubiquitous, for hardware and software developers to treat program interfaces as trade secrets, available only under restrictive licensing terms. This was partly aimed to prevent competitors or prospective licensees from taking advantage of open access to interfaces. A competitor's reimplementation of interfaces of a popular platform could enable it to build a system capable of running programs developed for the original platform and divert prospective customers from that platform.[31] Platform developers also made money by licensing interface information and often used license restrictions to prevent application developers from porting their software to other platforms.[32] Yet, customer demand for cross-platform compatibility and industry interest in writing programs that could connect to different systems had begun to grow as well.

B. Anti-Compatibility Arguments in Copyright Cases

Two decisions rendered by the Third Circuit Court of Appeals between 1983 and 1986 cast doubt on whether second comers could achieve software compatibility free from copyright liability. The first was *Apple Computer, Inc. v. Franklin Computer Corp.*[33] The second was *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*[34]

Apple introduced its Apple II personal computers to the market in 1977, along with numerous application programs designed to run on that platform. By 1983 Apple had sold more than 400,000 of these machines.[35] The

30. *But see* Samuelson, *supra* note 27, at 727–49 (discussing copyright's longstanding rule excluding functional creations from copyright protection). The *Google* decision noted that the CONTU Report concluded that "copyright 'should not grant anyone more economic power than is necessary to achieve the incentive to create.'" Google LLC v Oracle Am., Inc., 141 S. Ct. 1183, 1198 (2021) (quoting CONTU REPORT, *supra* note 26, at 12).

31. In the 1980s, IBM and Fujitsu engaged in a lengthy legal battle over Fujitsu's development of OS software that enabled its computers to be compatible with IBM mainframes. *See, e.g.*, Anita Stork, *The Use of Arbitration in Copyright Disputes:* IBM v. Fujitsu, 3 HIGH TECH. L.J. 241 (1988) (examining the rapid growth of IBM's base and Fujitsu's decision to develop IBM-compatible equipment and software).

32. Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992), discussed *infra* notes 84–96, was such a case. Accolade considered, but decided against, entering into a license to get access to the Sega platform interface information because of the exclusivity Sega demanded. *Accolade*, 977 F.2d at 1514.

33. 714 F.2d 1240 (3d Cir. 1983), discussed in BAND & KATOH, *supra* note 21, at 84–91.

34. 797 F.2d 1222 (3d Cir. 1986).

35. *Franklin*, 714 F.2d at 1242.

popularity of Apple II computers induced many independent software firms to develop programs to interoperate with that platform, which increased the value and attractiveness of the platform for consumers. Franklin, which had sold fewer than 1,000 of its ACE computers, recognized that it would be able to sell many more computers if its machines were compatible with Apple II.[36] To enable this compatibility, Franklin installed copies of Apple's OS programs on its ACE computers.

Apple sued Franklin for infringing its copyrights in the OS programs.[37] Franklin tried to justify this copying by claiming, among other things, that this was the only way it could achieve total compatibility with programs developed for the Apple II platform; hence, the expression of the Apple programs had merged with its ideas.[38] Franklin also argued that the Apple OS programs were unprotectable processes or methods of operation excluded from copyright protection under 17 U.S.C. § 102(b).[39]

The Third Circuit Court of Appeals found Franklin's merger argument unpersuasive: "Franklin may wish to achieve total compatibility with independently developed application programs written for the Apple II, but that is a commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expressions have merged."[40] Equally unavailing was Franklin's process/method exclusion argument. The Third Circuit characterized § 102(b) as merely a restatement of the idea/expression distinction.[41] Giving credence to Franklin's argument, the court reasoned, would make all programs uncopyrightable, contrary to Congressional intent.[42]

Because Apple had shown a likelihood of success on the merits and would likely suffer irreparable harm unless an injunction issued, the Third Circuit thought a preliminary injunction should issue against Franklin's further installation of the Apple II OS programs on its ACE machines.[43] Notably, though, Franklin's illegal act was not in reimplementing Apple interfaces in independent code to achieve compatibility, but rather the outright copying of the Apple OS. Franklin didn't even try to develop its own implementation of the OS programs. Yet, the court's anti-compatibility dicta

36. *Id.* at 1243.

37. *Id.* at 1244. Franklin's OS programs were virtually identical to Apple's. *Id.* at 1245.

38. *Id.* at 1245, 1253.

39. "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work." 17 U.S.C. § 102(b).

40. *Franklin*, 714 F.2d at 1253. *Oracle III* quoted this Third Circuit dicta favorably when dismissing the relevance of compatibility considerations to the copyrightability of the Java API elements that Google used in Android. *Oracle III*, 750 F.3d 1139, 1371 (Fed. Cir. 2014).

41. *Franklin*, 714 F.2d at 1252–53.

42. *Id.* at 1251–54.

43. *Id.* at 1253–55.

and dismissive interpretation of § 102(b)'s statutory exclusions of methods and processes cast doubt on future interoperability defenses.

A second key case on which anti-compatibility proponents relied was *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*[44] Although *Whelan* was not a compatibility case, the Third Circuit's conception of computer programs as needing a broad scope of copyright protection to preserve adequate incentives to develop software and its novel test for software copyright infringement provided the intellectual framework for those contending that program interfaces are protectable expressions.[45]

Jaslow had hired Whelan to develop a computer program to automate common dental lab business operations.[46] Initially, they intended to commercialize the Dentalab program as partners. After a falling out, Jaslow decided to develop his own dental lab business program, which was written in a different computer language for a different computer platform. Whelan alleged that Jaslow's program infringed copyright because he'd copied the "structure, sequence, and organization" (SSO) of her program.[47]

Whelan was the first appellate court decision to address whether nonliteral elements of computer programs, such as SSO and "look and feel," were protectable by copyright law.[48] The Third Circuit reasoned that because computer programs are "literary works," the SSO they embody should be as much protectable expression as the SSO of novels, poems, and other literary works are. The court articulated a test for judging infringement of software copyrights: "the purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea."[49] Insofar as other alternatives existed for achieving that purpose or performing that function, the plaintiffs' choices should be regarded as program expressions.[50] Because it was not necessary for Jaslow to use the same file and data structures as Whelan or the same SSO for certain subroutines, the court concluded there was no merger of idea and expression. As in *Franklin*, the Third Circuit regarded § 102(b) to be a restatement of the idea/expression distinction, finding it unnecessary to consider whether the subroutine SSO, for example, was an unprotectable method or process.[51] Consequently, the court affirmed the ruling that

44. 797 F.2d 1222 (3d Cir. 1986).

45. *Id.* at 1236–37. Computer Associates and its amici relied on *Whelan* in the *Altai* case discussed *infra* text accompanying notes 53–81.

46. The facts and procedural history of the *Whelan* case are set forth at 797 F.2d at 1225–29.

47. *Id.* at 1224 n.1 (internal quotation marks omitted) (adopting SSO terminology).

48. *Id. See also id.* at 1231 (noting the creativity required to design program look and feel and that "[b]y far the larger portion of the expense and difficulty in creating computer programs is attributable to the development of the structure and logic of the program," not to its coding).

49. *Id.* at 1236 (emphasis omitted).

50. *Id.* at 1236 & n.28.

51. *Id.* at 1234–36.

Jaslow's program infringed Whelan's copyright.[52] Under the *Whelan* decision's proposed test for software copyright infringement, along with its narrow interpretation of the merger doctrine and of § 102(b)'s exclusions, software would have been given the most expansive possible scope of copyright protection.

C. The Pro-Compatibility Decisions

1. Altai.—The first appellate court to directly address whether interfaces of computer programs needed to achieve compatibility with other programs were within the scope of software copyright protection was the Second Circuit Court of Appeals in its 1992 decision *Computer Associates Int'l, Inc. v. Altai, Inc.*[53]

Computer Associates (CA) and Altai were competitors in the market for scheduling programs designed to run on IBM mainframes. Altai hired a former CA engineer to work on the development of Zeke, a compatibility subprogram designed to make it easier for Altai to update its OSCAR scheduling program so it could exchange information with various IBM OS programs. Unbeknownst to Altai's management, that engineer copied a substantial part of CA's code in Zeke.[54] After learning of this infringement, Altai's management provided a new team of engineers with a list of services and parameter lists from a "clean room" process and directed them to reimplement these elements in noninfringing code.[55] Altai admitted infringement as to the copied code, but contended that the revised Zeke was noninfringing.[56] CA argued that Zeke still infringed because nonliteral structural elements of Zeke were substantially similar to and an infringement of CA's Adapter program copyright.[57] A trial court ruled against CA's claim because similarities in the two programs were mainly due to constraints

52. *Id.* at 1248.

53. 982 F.2d 693 (2d Cir. 1992). Compatibility defenses had succeeded in two earlier district court decisions. *See* Secure Servs. Tech., Inc. v. Time & Space Processing, Inc., 722 F. Supp. 1354 (E.D. Va. 1989) (finding that design of software for secure fax machines for sale to U.S. military was constrained by T-30 protocol); NEC Corp. v. Intel Corp., No. C-84-20799-WPG, 1989 WL 67434 (N.D. Cal. Feb. 6, 1989), (finding that hardware constrained design choices for developing compatible microcode). In 1989 a group of ten intellectual property scholars reached consensus that the *Franklin* dicta about program compatibility being irrelevant to copyrightability was not consistent with traditional principles of copyright law. *See* Donald S. Chisum, Rochelle Cooper Dreyfuss, Paul Goldstein, Robert A. Gorman, Dennis S. Karjala, Edmund W. Kitch, Peter S. Menell, Leo J. Raskind, Jerome H. Reichman & Pamela Samuelson, *LaST Frontier Conference Report on Copyright Protection of Computer Software*, 30 JURIMETRICS J. 15, 19 (1989).

54. *Altai*, 982 F.2d at 699–700. For an excellent detailed review of the *Altai* litigation, see BAND & KATOH, *supra* note 21, at 112–30.

55. *Altai*, 982 F.2d at 700.

56. *Id.* at 701.

57. *Id.* at 702.

imposed by the need to be compatible with and provide services to the IBM OS programs.[58]

The Second Circuit affirmed the trial court's ruling in favor of Altai.[59] It based its framework on a more expansive view of copyright's idea/expression distinction. Developed by the Supreme Court approximately 150 years ago in *Baker v. Selden*,[60] and since codified,[61] the idea/expression distinction ensures that copyright law protects only the original expression of an idea, method, or system, and not the idea, method, or system itself, or any "necessary incidents" to that idea, method, or system.[62] *Altai* articulated a new "Abstraction–Filtration–Comparison" (AFC) test to apply that distinction to software copyright infringement.[63] The first step of this test calls for constructing a hierarchy of abstractions for the plaintiff's program. The court articulated six layers in its program abstraction hierarchy from most abstract (the general purpose or function of the program) to the most concrete and specific (source and object code).[64] The second step calls for filtering out unprotectable elements to ensure that the third and final

58. Comput. Assocs. Int'l, Inc. v. Altai, Inc., 775 F. Supp. 544, 562 (E.D.N.Y. 1991). The district court criticized and declined to follow *Whelan*'s overbroad test for software copyright infringement. *Id.* at 558–60. The Software Publishers Association (SPA) filed an amicus curiae brief in support of CA's appeal to the Second Circuit, criticizing the district court's *Altai* opinion for failing to follow *Whelan*. *See* Brief Amicus Curiae of the Software Publishers Association, *Altai*, 982 F.2d 693 (No. 91-7893), 1991 WL 11010230. The main goal of the SPA brief was to persuade the Second Circuit to adopt the *Whelan* approach to protecting program structures. *Id.* at 7–8. The SPA brief did not directly argue that program interfaces were protectable by copyright law. However, it characterized parameter lists, which set forth interface elements, as part of the SSO that copyright law protects in software just as in other literary works. *Id.* at 8. But see Brief Amicus Curiae of Compaq Computer Corp., Novell, Inc., and Borland Int'l, Inc., *Altai*, 982 F.2d 693 (No. 91-7893), 1992 WL 12013079, for an amicus brief filed by two of the larger members of SPA and one of its former members strongly objecting to the SPA's endorsement of *Whelan*.

59. *Altai*, 982 F.2d at 714–15. In support of Altai on the compatibility issue was the Brief Amicus Curiae of Am. Comm. for Interoperable Sys., *Altai*, 982 F.2d 693 (No. 91-7893), 1991 WL 11010231.

60. 101 U.S. 99 (1879).

61. 17 U.S.C. § 102(b).

62. *Baker*, 101 U.S. at 103. *Baker* is often credited as the origin of the merger doctrine because of its statement that "necessary incidents" to methods and other useful arts are not within the scope of copyright protection. *But see* Samuelson, *supra* note 14, 419–25 ("While the holding in *Baker* is consistent with the merger doctrine, *Baker* cannot fairly be said to have given birth to it."). In this respect, the merger doctrine is best understood as a corollary to § 102(b)'s exclusion of ideas, methods, and systems. *Id.* at 451–53.

63. *Altai*, 982 F.2d at 706–11. The court rejected the Third Circuit's *Whelan* test for software copyright infringement under which nonliteral elements of programs were protectable unless there was only one way to structure those aspects of the programs. *Id.* at 705–06. The *Altai* AFC test for software copyright infringement has been followed and cited approvingly by every subsequent appellate court to have considered it. *See infra* note 82 (discussing subsequent cases).

64. *Id.* at 706–07. By directing the construction of abstractions hierarchies when analyzing nonliteral software copyright infringement claims, the Second Circuit rejected *Whelan*'s flattened conception of the protectability of all nonliteral elements unless no alternatives were available.

comparison step focuses on similarities in the two programs' expressions, not on similarities in unprotectable elements.[65]

Altai identified three categories of unprotectable elements that must be filtered out before proceeding to the comparison step: first, those that are dictated by efficiency considerations; second, those that are dictated by external factors; and third, those that are in the public domain.[66] It identified several subcategories of elements that it viewed as dictated by external factors.[67] The court found no error in the district court's conclusion that similarities in the CA and Altai parameter lists were largely dictated by functional considerations or were in the public domain.[68] The lists of services were, moreover, "dictated by the nature of other programs with which [they were] designed to interact and, thus, [are] not protected by copyright."[69] The *Altai* decision is particularly notable because, unlike *Franklin*, it suggested that similarities driven by a competitor's desire for compatibility were functional and therefore not protectable.

Although the Second Circuit invoked copyright's *scenes a faire* doctrine as the basis for characterizing these five types of unprotectable elements, it twice spoke of these elements as "dictated by external factors."[70] "Dictated by" is the language of the merger doctrine, not the *scenes a faire* doctrine.[71]

Nonetheless, the Second Circuit agreed with the lower court that the similarities between CA's and Altai's scheduling programs should be filtered out when necessary to be compatible with other programs, when dictated by functional demands of those programs, or when elements are in the public domain.[72] Hence, the court affirmed the finding in Altai's favor on the copyright claim. It concluded that compatibility was a practical programming and business necessity that could not be protected by copyright. Specifically, the court noted:

65. *Id.* at 707–10.

66. *Id.* at 707. One of us has noted elsewhere that the Second Circuit failed to direct filtration of procedures, processes, systems, and methods of operation in keeping with 17 U.S.C. § 102(b). Pamela Samuelson, *Functionality and Expression in Computer Programs: Refining the Tests for Software Copyright Infringement*, 31 BERKELEY TECH. L.J. 1215, 1235–37 (2016). This failure is odd because the legislative history reveals that Congress added these exclusions to the copyright act to ensure that the scope of copyright protection for computer programs would not be construed too broadly. H.R. REP. NO. 94-1476, at 57 (1976); S. REP. NO. 94-473, at 54 (1975). Subsequent cases have adapted the *Altai* AFC test to filter out unprotectable methods and processes. *See, e.g.*, Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 836–37, 845 (10th Cir. 1993) (holding that processes are not protectable under § 102(b) and must be filtered out).

67. *Altai*, 982 F.2d at 709–10.

68. *Id.* at 715.

69. *Id.*

70. *Id.* at 709–10.

71. *See, e.g.*, Samuelson, *supra* note 14, at 447–50 (distinguishing the merger and *scenes a faire* doctrines).

72. *Altai*, 982 F.2d at 714–15.

> "[I]n many instances it is virtually impossible to write a program to perform particular functions in a specific computing environment without employing standard techniques." 3 NIMMER § 13.03[F][3], at 13-65. This is a result of the fact that a programmer's freedom of design choice is often circumscribed by extrinsic considerations such as (1) the mechanical specifications of the computer on which a particular program is intended to run; (2) *compatibility requirements of other programs with which a program is designed to operate in conjunction*; (3) computer manufacturers' design standards; (4) demands of the industry being serviced; and (5) *widely accepted programming practices within the computer industry*.
>
> . . . .
>
> . . . [T]he overlap exhibited between the list of services required for both ADAPTER and OSCAR 3.5 was "determined by the demands of the operating system and of the applications program to which it [was] to be linked through ADAPTER or OSCAR. . . ." [*Comput. Assocs. Int'l, Inc. v. Altai, Inc.*, 775 F. Supp. 544, 562 (E.D.N.Y. 1991).] In other words, this aspect of the program's structure was dictated by the nature of other programs with which it was designed to interact and, thus, is not protected by copyright.[73]

The *Altai* decision was notable for several other reasons. It soundly criticized *Whelan* for its overbroad software copyright infringement test, for its inaccurate understanding of computer science, and for its focus on metaphysical rather than practical considerations.[74] Unlike *Whelan*, the Second Circuit's *Altai* decision recognized that "[t]he essentially utilitarian nature of a computer program . . . complicates the task of distilling its idea from its expression."[75] In contrast to conventional literary and artistic works, "computer programs hover even more closely to the elusive boundary line described in § 102(b)."[76] Interpreting the scope of copyright protection for programs too broadly could result in anti-competitive effects.[77] The Second Circuit rejected CA's and its amici's argument that investments in software development would be harmed unless computer programs were accorded strong copyright protection. The court noted that *Whelan* had been decided before the Supreme Court's *Feist Publications, Inc. v. Rural Telephone Service Co.*[78] decision, which "teaches that substantial effort alone cannot

73. *Id.* at 709–10, 715 (emphasis added).

74. *Id.* at 705–06. The Second Circuit agreed with *Whelan* on the point that some nonliteral elements of computer programs could be copyright-protectable; however, it did not find SSO to be a helpful way to conceptualize protectable elements in programs. *See id.* at 706 (proposing AFC test instead).

75. *Id.* at 704.

76. *Id.*

77. *Id.* at 711.

78. 499 U.S. 340 (1991).

confer copyright status on an otherwise uncopyrightable work," and observed that incentive-based arguments for broad copyright protections would, if adopted, have "a corrosive effect on certain fundamental tenets of copyright doctrine."[79] As a result, it "may well be that the Copyright Act serves as a relatively weak barrier against public access to the theoretical interstices behind a program's source and object codes" given the intermixture of functionality and expression in programs.[80] The court conjectured that patent law might be a "more appropriate rubric" for IP protection for software, and even suggested that Congress should convene a CONTU II to consider copyright scope issues.[81]

2. Altai*'s Progeny.*—The *Altai* decision swiftly led to widespread rejections of *Whelan*, particularly in cases involving compatibility.[82] Within months after the *Altai* decision, the Ninth Circuit concurred in its conclusion that software developers should be able to achieve compatibility free from copyright liability in another landmark software copyright case, this one involving copyright's fair use doctrine. That doctrine permits certain uses that would otherwise infringe copyright, particularly if they serve a transformative or other socially useful purpose or don't cause market harm.[83] In *Sega Enterprises, Ltd. v. Accolade, Inc.*,[84] the Ninth Circuit echoed the *Altai* decision in its characterization of computer programs as utilitarian works that "contain many logical, structural, and visual display elements that are dictated by the function to be performed, by considerations of efficiency, or by external factors such as compatibility requirements and industry

79. *Altai*, 982 F.2d at 711–12 (citing Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 111 S. Ct. 1282, 1295 (1991)).

80. *Id.* at 712.

81. *Id.* It further noted that many of the software copyright decisions "reflect the courts' attempt to fit the proverbial square peg in a round hole." *Id.*

82. In addition to the several decisions discussed in this section, we note that the Tenth Circuit followed the *Altai* decision and adopted its test and rationale in Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 836–37, 845 (10th Cir. 1993) as did the Fifth Circuit in Eng'g Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1342–44 (5th Cir. 1994), *clarified,* 46 F.3d 408 (5th Cir. 1995). Thus, by 1995, *Altai* had become the leading software copyright case. *See, e.g.*, Mark A. Lemley, *Convergence in the Law of Software Copyright?*, 10 HIGH TECH. L.J. 1, 14–15 (1995) (discussing *Altai*'s endorsement by multiple circuit courts in the United States and by courts in other countries). This continues to be true. *See* Pamela Samuelson, *A Fresh Look at Tests for Nonliteral Copyright Infringement*, 107 NW. U. L. REV. 1821, 1838–39, 1838 n.108 (2013) (noting that as of Feb. 8, 2013, the *Altai* decision had been cited 236 times and by courts in every circuit except the D.C. Circuit).

83. 17 U.S.C. § 107. The test for fair use is multi-factor and fact-specific, and a full discussion is beyond the scope of this Article.

84. 977 F.2d 1510 (9th Cir. 1992).

demands."[85] These considerations meant, said the Ninth Circuit, that the scope of copyright protection for computer programs was thinner than for other types of literary works.[86]

The main legal question in *Accolade* was whether the defendant had infringed copyright by making copies of Sega programs in the course of reverse engineering that code in order to get access to information necessary for Accolade to adapt its videogames so they could run on the Sega Genesis platform.[87] The Ninth Circuit decided that Accolade had a legitimate purpose in making these intermediate copies because this was the only way that Accolade could get access to that information.[88] This consideration favored Accolade's fair use defense because the Sega interface constituted the functional requirements for achieving compatibility, which the court viewed as unprotectable "procedures" that were excluded from copyright protection under § 102(b).[89] To rule that Accolade's reverse-engineering copies were infringements would give Sega "a *de facto* monopoly" over these functional elements of its programs.[90] Thus, while *Sega* was decided on fair use grounds, its fair use analysis depends heavily on the underlying conclusion that interfaces were unprotectable procedures.

The *Accolade* decision also cited *Altai* and quoted from the CONTU Commission's report for the proposition that "when specific instructions,

85. *Id.* at 1524. The Ninth Circuit said it was "in wholehearted agreement" with *Altai* about software copyright decisions as akin to square pegs in round holes. *Id.* It also cited approvingly to *Altai*'s repudiation of the *Whelan* test for software copyright infringement. *Id.* at 1525. Although the Ninth Circuit's *Accolade* decision was influenced by *Altai*, that circuit has not adopted the *Altai* AFC test as such when reviewing software copyright decisions. Yet, like the Second Circuit, the Ninth Circuit engages in filtration of unprotectable elements. *See* Apple Comput., Inc. v. Microsoft Corp., 35 F.3d 1435, 1438–39 (9th Cir. 1994) (approving of a trial court's application of "limiting doctrines of originality, functionality, standardization, *scenes a faire*, and merger," as well as upholding its filtration of user interface processes).

86. *Accolade*, 977 F.2d at 1526.

87. *Id.* at 1513–15. The American Committee for Interoperable Systems (ACIS) filed an amicus curiae brief in support of Accolade, just as it had in support of Altai. Brief Amicus Curiae of Am. Comm. for Interoperable Sys., *Accolade*, 977 F.2d 1510 (No. 3:91-CV-03871).

88. *Accolade*, 977 F.2d at 1514, 1525–26. *See also* Atari Games Corp. v. Nintendo of America, Inc., 975 F.2d 832, 843–44 (Fed. Cir. 1992) (asserting that intermediate copying can be a fair use when it is necessary "to understand the ideas and processes in a copyrighted work") (approvingly cited in *Accolade*, 977 F.2d at 1525). The Federal Circuit correctly anticipated that the Ninth Circuit would hold that making copies of computer programs to get access to information necessary to achieving interoperability would be fair use. *Id.* at 842–44. With ample citations to *Altai*, the Federal Circuit in *Atari Games* accepted that when assessing software copyright infringement claims, courts "must filter out as unprotectable the ideas, expression necessarily incident to the idea, expression already in the public domain, expression dictated by external factors (like the computer's mechanical specifications, compatibility with other programs, and demands of the industry served by the program), and expression not original to the programmer or author." *Id.* at 839.

89. *Accolade*, 977 F.2d at 1526. *See also* Sony Comput. Ent., Inc. v. Connectix Corp., 203 F.3d 596, 604 (9th Cir. 2000) (characterizing program interfaces as unprotectable procedures).

90. *Accolade*, 977 F.2d at 1526.

even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to infringement."[91] This statement, which plainly expresses the merger doctrine, was relevant in *Accolade* because the defendant's games would not run on the Sega platform unless they included a small segment of Sega code that caused a Sega trademark to pop up.[92]

Compatibility was at the heart of *Sega*'s fair use finding. Accolade wanted to make video games compatible with Sega's game console over Sega's objection. To make its games run on Sega's platform, Accolade copied the entirety of Sega's computer code in order to "reverse engineer" the code and extract only the APIs—the portions necessary to ensure compatibility. The Ninth Circuit held that was a fair use even though it involved intermediate copying of the entirety of the code, because making that intermediate copy was necessary to get access to the interface components—which the Ninth Circuit found to be "unprotected."[93] The court emphasized that "because Accolade has a legitimate interest in gaining such access (in order to determine how to make its cartridges compatible with the Genesis console)," its copying of the code to replicate the interface components was a fair use.[94]

The fact that Accolade sought to write its own original programs, not to copy Sega's programs, loomed large in the Ninth Circuit's analysis:

> Accolade copied Sega's software solely in order to discover the functional requirements for compatibility with the Genesis console—aspects of Sega's programs that are not protected by copyright. With respect to the video game programs contained in Accolade's game cartridges, there is no evidence in the record that Accolade sought to avoid performing its own creative work. Indeed, most of the games that Accolade released for use with the Genesis console were originally developed for other hardware systems. . . . [A]lthough Accolade's ultimate purpose was the release of Genesis-compatible games for sale, its direct purpose in copying Sega's code, and thus its direct use of the copyrighted material, was simply to study the functional requirements for Genesis compatibility so that it could modify existing games and make them usable with the Genesis console. . . . On these facts, we conclude that Accolade copied Sega's code for a legitimate, essentially non-exploitative purpose . . . .[95]

91. *Id.* at 1524 (first quoting CONTU REPORT, *supra* note 26, at 20; and then citing Comput. Assoc. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 708 (2d Cir. 1992)).

92. *Id.* at 1524 n.7.

93. *Id.*

94. *Id.* at 1520.

95. *Id.* at 1522–23 (citation omitted).

Nor was the court troubled that Accolade engaged in verbatim copying of some program interfaces in order to achieve that legitimate compatibility purpose:

> [C]omputer programs are, in essence, utilitarian articles—articles that accomplish tasks. As such, they contain many logical, structural, and visual display elements that are dictated by the function to be performed, by considerations of efficiency, or by external factors such as compatibility requirements and industry demands. . . . "[W]hen specific instructions, even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to infringement."[96]

Thus, while *Sega* was a fair use case, the reason the court relied on fair use was that Accolade copied all of Sega's code, not just the APIs. The Ninth Circuit clearly treated the APIs themselves as uncopyrightable, and indeed found that the copying of protected code was justified because it gave Accolade access to the unprotected APIs.

In *Sony Computer Entertainment, Inc. v. Connectix Corp.*,[97] the Ninth Circuit went still further, holding that it was fair use to create an emulator of the Sony game console—copying the code not just to reverse engineer it but to test how programs worked with it—because the purpose was to produce a new product that worked with the old system:

> We find that Connectix's Virtual Game Station is modestly transformative. The product creates a new platform, the personal computer, on which consumers can play games designed for the Sony PlayStation. This innovation affords opportunities for game play in new environments, specifically anywhere a Sony PlayStation console and television are not available, but a computer with a CD-ROM drive is. More important, the Virtual Game Station itself is a wholly new product, notwithstanding the similarity of uses and functions between the Sony PlayStation and the Virtual Game Station.[98]

The fact that Sony might lose sales to the Connectix system did not militate against fair use on the fourth factor. "[B]ecause the Virtual Game Station is transformative, and does not merely supplant the PlayStation console, the Virtual Game Station is a legitimate competitor in the market for platforms on which Sony and Sony-licensed games can be played," so the loss of market share was not attributable to copyright infringement, but to legitimate competition.[99] The Ninth Circuit reversed the grant of a preliminary injunction against Connectix's emulator.

96. *Id.* at 1524 (citations omitted) (quoting CONTU Report, *supra* note 26, at 20).
97. 203 F.3d 596 (9th Cir. 2000).
98. *Id.* at 606.
99. *Id.* at 607.

The Eleventh Circuit joined the *Altai*-inspired compatibility bandwagon in *Bateman v. Mnemonics, Inc.*[100] Mnemonics's main business was licensing of an application program that automated parking garage operations. Mnemonics initially developed its software to run on Bateman's OS. After Bateman terminated that OS license, Mnemonics decided to develop an alternative OS so it could stay in business and continue to commercialize its software.[101] Bateman argued that Mnemonics could not raise an *Altai* compatibility challenge because Bateman thought the *Altai* filtration step applied only to nonliteral elements of computer programs, not to literal copying of some Bateman code.[102] The trial court agreed and accordingly instructed the jury, which rendered a verdict of infringement for Bateman.

On appeal, Mnemonics argued that interface specifications of computer programs are uncopyrightable as a matter of law, and alternatively, that some copying of Bateman's code was justifiable if necessary to achieve compatibility.[103] The Eleventh Circuit rejected the first argument, but found the second argument persuasive; it consequently overturned the verdict because of the trial judge's erroneous jury instructions.[104] The *Bateman* decision cited approvingly to *Altai* and *Accolade* in support of this holding. The court agreed with Mnemonics that it should not be legally obliged to rewrite its application program so that it would run on an entirely different OS program.[105]

The Eleventh Circuit was, however, agnostic about "[w]hether [copyright] protection is unavailable because these [external constraint] factors render the expression unoriginal, nonexpressive per 17 U.S.C. § 102(b), or whether these factors compel a finding of fair use, copyright estoppel, or misuse" because whatever the legal doctrine, "the result is to deny copyright protection to portions of the computer program."[106] Although the Eleventh Circuit did not specify which doctrine was most applicable when some copying of code was "dictated by compatibility requirements,"[107] its "dictated by" language evokes the doctrine of merger.

Like the Ninth Circuit in *Accolade* and the Eleventh Circuit in *Bateman*, the Sixth Circuit in *Lexmark International, Inc. v. Static Control*

100. 79 F.3d 1532, 1547 (11th Cir. 1996).

101. *Id.* at 1538–40.

102. *Id.* at 1543–45.

103. *Id.* at 1547. As in *Altai* and *Accolade*, ACIS filed a brief in support of Mnemonics' appeal of a jury finding of infringement on the uncopyrightability of program interfaces. Brief Amicus Curiae of Am. Comm. for Interoperable Sys., *Bateman*, 79 F.3d 1532 (No. 93-3234), 1994 WL 16129974.

104. *Bateman*, 79 F.3d at 1547.

105. *Id.* at 1547 n.31.

106. *Id.* at 1547.

107. *Id.*

Components, Inc.[108] decided that exact copying of another developer's program does not infringe when necessary for interoperability.[109] Unlike the plaintiffs in *Altai*, *Accolade*, and *Bateman*, Lexmark's primary business was not the development of software, but rather the manufacture and sale of printers and printer cartridges. In an effort to thwart competitors from making and selling printer cartridges capable of running in its printers, Lexmark embedded programs in its printers and printer cartridges that exchanged authentication messages so that only Lexmark's printer cartridges would work in Lexmark printers.

Static's primary business was manufacturing and selling semiconductor chips. Some of its customers wanted to sell printer cartridges that would work with Lexmark printers. To enable this, Static installed copies of the Lexmark printer cartridge software on its chips. Lexmark then sued Static for copyright infringement and persuaded a trial court to issue a preliminary injunction forbidding Static to reproduce copies of the Lexmark printer cartridge program.[110]

The Sixth Circuit characterized the Lexmark programs as "lock-out" codes that "fall on the functional-idea rather than the original-expression side of the copyright line."[111] But it did not apply the pure AFC test from *Altai*. Rather, the court invoked two other copyright doctrines that flow from the idea/expression distinction to protect compatibility. The first, the *scenes a faire* doctrine, makes clear that § 102(b) treats as unprotectable not only the idea itself, but various stock techniques, concepts, and building blocks one might use in implementing that idea.[112] The other, the merger doctrine, extends the idea/expression limitation to deny protection to a work altogether when there are only a limited number of ways of implementing that idea.[113]

108. 387 F.3d 522 (6th Cir. 2004).

109. *Id.* at 536. CCIA filed an amicus curiae brief in support of Static Controls. Brief Amicus Curiae of Comput. & Commc'ns Indus. Ass'ns in Support of Static Control Components, Inc., *Lexmark*, 387 F.3d 522 (No. 03-5400), 2003 WL 22318988.

110. *Lexmark*, 387 F.3d at 531–32. The trial court also found that Lexmark was likely to succeed with its claim that Static was liable for circumvention of technical measures under 17 U.S.C. § 1201 because the copied code bypassed the authentication sequence between the printer and printer cartridge software. *Id.* at 532. The Sixth Circuit was unconvinced that Lexmark would succeed on that claim. *Id.* at 545–50.

111. *Id.* at 536.

112. *See, e.g.*, Atari, Inc. v. N. Am. Philips Consumer Elec. Corp., 672 F.2d 607, 616 (7th Cir. 1982) (defining *scenes a faire* as "incidents, characters or settings which are as a practical matter indispensable, or at least standard, in the treatment of a given topic") (quoting Alexander v. Haley, 460 F. Supp. 40, 45 (S.D.N.Y. 1978)); *see also* Leslie A. Kurtz, *Copyright: The* Scenes a Faire *Doctrine*, 41 FLA. L. REV. 79, 80 (1989) (tracing this doctrine to Cain v. Universal Pictures Co., 47 F. Supp. 1013, 1017 (S.D. Cal. 1942), in which a movie had allegedly copied a scene from plaintiff's novel).

113. Morrissey v. Procter & Gamble Co., 379 F.2d 675, 678 (1st Cir. 1967) ("When the uncopyrightable subject matter is very narrow, so that 'the topic necessarily requires,' if not only

When "compatibility requires that a particular code sequence be included in the component device to permit its use, the merger and *scenes a faire* doctrines generally preclude the code sequence from obtaining copyright protection."[114] The Sixth Circuit held that the trial court had erred in refusing to consider whether external factors such as compatibility requirements had narrowed the range of options available to Static under the merger and *scenes a faire* doctrines.[115]

Although Lexmark's expert testified that Static could have written a new interoperable printer-cartridge program instead of copying Lexmark's program,[116] Static's expert testified that the alternatives identified by Lexmark's expert were "trivial" or "so inefficient and repetitive as to be 'ridiculous.'"[117] Moreover, Static's expert offered "unchallenged testimony that it would be 'computationally impossible' to modify the checksum value" that the Lexmark printer program sent to the printer-cartridge program.[118] In view of this evidence, the Sixth Circuit concluded that the merger and *scenes a faire* doctrines precluded Lexmark from claiming that Static infringed the printer-cartridge program.[119]

Seemingly out of an abundance of caution, the Sixth Circuit decided to address Static's fair use defense as well.[120] Static had not, the court noted, copied the Lexmark code to exploit its commercial value as a copyrighted work, but rather for a different purpose: namely, to enable printer cartridges to work in Lexmark printers.[121] Nor was Static's use of the Lexmark code harming the marketability of that program. This use may have harmed Lexmark's market for selling printer cartridges, but this was not the type of market harm that copyright law aims to prevent.[122]

one form of expression, at best only a limited number, to permit copyrighting would mean that a party or parties, by copyrighting a mere handful of forms, could exhaust all possibilities of future use of the substance." (citations omitted)).

114. *Lexmark*, 387 F.3d at 536.

115. *Id.* at 537–38.

116. *Id.* at 539.

117. *Id.* at 540.

118. *Id.* at 542.

119. *Id.* at 535, 540–42.

120. *Id.* at 544–45. The *Google* decision cites approvingly to *Lexmark*'s fair use analysis. Google LLC v. Oracle Am., Inc., 141 S. Ct. 1183, 1198 (2021). While the Sixth Circuit in *Lexmark* addressed the fair use defense out of an abundance of caution, 387 F.3d at 544–45, the court relied on merger and *scenes a faire* more than fair use as the basis for ruling against Lexmark's copyright claim. *Id.* at 534–44.

121. *Lexmark*, 387 F.3d at 544.

122. *Id.* at 545.

A final appellate court case from this period is *Mitel, Inc. v. Iqtel, Inc.*[123] To activate and manipulate features of its telephone-call-controller system, Mitel developed a set of more than sixty command codes. When Iqtel entered the call-controller market, it decided it could successfully compete with Mitel only if its call controllers were compatible with Mitel's. To enable compatibility, Iqtel programmed its call controllers to accept Mitel's command codes and transform them into corresponding Iqtel codes.[124] Mitel claimed that this copying of its codes infringed copyright.

The Tenth Circuit upheld the lower court's denial of a preliminary injunction motion in part because "much of the expression in Mitel's command codes was dictated by the proclivities of technicians and limited by significant hardware, compatibility, and industry requirements."[125] Mitel's product-management team had, for instance, adopted many of the values in the Mitel codes "in response to customer demand or to ensure compatibility with equipment already installed in the central offices of Mitel's customers."[126] Others were "dictated by the limits inherent in the public telephone networks that the call controllers accessed."[127] The court relied on the *scenes a faire* doctrine as the basis for ruling these elements of the Mitel command codes were unprotectable by copyright law.[128] And like the Ninth Circuit in *Connectix*, the court did not treat the fact that the defendant's goal in producing a compatible product was to compete with the plaintiff as a reason to extend copyright protection to the command codes. To

123. 124 F.3d 1366 (10th Cir. 1997). Call controllers are hardware devices that automate various telephone utilities such as the selection of particular long-distance carriers. *Id.* at 1368. Similarities in software-application programs due to common elements have more generally been held unprotectable under the *scenes a faire* doctrine. *See, e.g.*, Hutchins v. Zoll Med. Corp., 492 F.3d 1377, 1384–85 (Fed. Cir. 2007) (holding no copyright in common phrases for computerized system for performing cardiopulmonary-resuscitation procedures); Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1472–74, 1478 (9th Cir. 1992) (holding no copyright in user-interface similarities common to outlining programs).

124. *Mitel*, 124 F.3d at 1368–69.

125. *Id.* at 1375. *See also* Plains Cotton Coop. Assoc. of Lubbock v. Goodpasture Serv., Inc., 807 F.2d 1256, 1262 (5th Cir. 1987) (declining to extend copyright protection to elements of a program's user interface that were dictated by "the externalities of the cotton market" and other "market factors"); Auto Inspection Servs., Inc. v. Flint Auto Auction, Inc., No. 06-15100, 2006 WL 3500868, at *3, 7 (E.D. Mich. Dec. 4, 2006) (stating that the software's onscreen displays were written to "conform to all industry requirements and follow the normal and logical flow of a vehicle inspection").

126. *Mitel*, 124 F.3d at 1375.

127. *Id.*

128. *Id.* at 1374–76. The court regarded other elements of the Mitel command codes as lacking originality. *Id.* at 1373–74. *See also* Assessment Techs. of WI, LLC v. WIREdata, Inc., 350 F.3d 640, 644–45 (7th Cir. 2003) (characterizing the assertion of a copyright infringement claim based on extraction of unprotectable data from a database as misuse).

the contrary, customer demands for compatible products were a reason that justified copying interface components.[129]

3. Lotus v. Borland *and Methods of Operation.*—The interests of users in the enablement of compatibility also drove the result in *Lotus Development Corp. v. Borland International, Inc.*[130] Lotus sued Borland for copyright infringement because its competing spreadsheet program reproduced the Lotus command hierarchy in its "[e]mulation" user interface.[131] The goal was to enable users who had constructed macros in the Lotus 1-2-3 macro language to execute them on Borland's platform and to take advantage of the users' investments in learning the Lotus 1-2-3 command hierarchy.[132]

Borland's principal defense was that the command hierarchy was an integral part of the Lotus macro system or method of operation, which was unprotectable by copyright law under § 102(b). Borland argued this was closely akin to the selection and arrangement of columns and headings for a bookkeeping system which the Supreme Court had held unprotectable in *Baker v. Selden*.[133]

Like the Third Circuit in *Franklin* and *Whelan*, the district court in *Borland* regarded § 102(b) as excluding from copyright protection only abstract elements of protected works.[134] Because Lotus could have arranged its spreadsheet commands in a large number of ways, the particular way it had selected and arranged them was, in the district court's view, protectable expression.[135] That court found Borland's copying of them to infringe.[136]

On appeal, Borland emphasized that its emulation interface had to embody the 1-2-3 command hierarchy using exactly the same commands in exactly the same order to enable user macros to be executed on Borland's platform in support of its § 102(b) challenge to the copyrightability of the

129. *Mitel*, 124 F.3d at 1375–76. One of us has explained why compatibility favors a finding of fair use of APIs, including the utilitarian nature of APIs and a misunderstanding of the role of "good faith" in fair use. Samuelson & Asay, *supra* note 16, at 544–61.

130. 49 F.3d 807 (1st Cir. 1995), *aff'd by an equally divided court*, 516 U.S. 233 (1995) (per curiam). ACIS filed amicus curiae briefs in support of Borland with the First Circuit and with the Supreme Court. Brief Amicus Curiae of Am. Comm. for Interoperable Sys., *Borland*, 49 F.3d 807 (No. 93-2214), 1993 WL 13624510 [hereinafter ACIS Borland Amicus]; Brief Amici Curiae of Am. Comm. for Interoperable Sys. and Comput. & Commc'ns Indus. Ass'n in Support of Respondent, *Borland*, 516 U.S. 233 (1996) (No. 94-2003), 1995 WL 728487 [hereinafter ACIS-CCIA Borland Amicus].

131. *Borland*, 49 F.3d at 810.

132. *Id.*

133. *Id.* at 813–14. The First Circuit did not find *Baker* to be as closely analogous as Borland asserted. *Id.* at 814.

134. Lotus Dev. Corp. v. Borland Int'l, Inc., 799 F. Supp. 203, 216–17 (D. Mass. 1992).

135. *Id.* at 217–18.

136. *Id.* at 223.

Lotus command hierarchy.[137] The First Circuit found "absurd" Lotus's theory that users should have to learn new sets of commands to perform the same functions on a different software platform.[138] "The fact that there may be many different ways to operate a computer program . . . using a set of hierarchically arranged command terms[] does not make the actual method of operation chosen copyrightable; it still functions as a method for operating the computer and as such is uncopyrightable."[139]

The First Circuit also balked at the idea that users should have to rewrite their macros using a different command hierarchy. "We think that forcing the user to cause the computer to perform the same operation in a different way ignores Congress's direction in § 102(b) that 'methods of operation' are not copyrightable."[140] But rather than rely directly on the idea/expression distinction or the merger and *scenes a faire* doctrines, the court turned to other language in § 102(b) that excludes from protection not only ideas, but also "system[s]" and "method[s] of operation."[141] Lotus's menu command hierarchy, the court said, was like the buttons on a VCR: the buttons are labeled, but the buttons themselves aren't copyrightable because they control the operation of the device.[142]

Judge Boudin's concurrence in *Borland* elaborated on these observations. He found it "very hard to see that Borland has shown any interest in the Lotus menu except as a fallback option for those users already committed to it by prior experience or in order to run their own macros using the 1-2-3 commands."[143] Judge Boudin observed that if Lotus could protect the command hierarchy through copyright, "users who have learned the command structure of Lotus 1-2-3 or devised their own macros are locked into Lotus, just as a typist who has learned the QWERTY keyboard would be captive of anyone who had a monopoly on the production of such a keyboard."[144] If Borland developed a superior program, "good reasons exist for freeing it to attract old Lotus customers: to enable the old customers to

137. *Borland*, 49 F.3d at 810, 815, 818. The Solicitor General's amicus curiae brief in response to Google's first Petition for Certiorari in the *Oracle* case suggested that the merger doctrine was a more persuasive rationale for the *Borland* decision. *See* Brief for the U.S. as Amicus Curiae at 20, Google, Inc. v. Oracle America, Inc., 576 U.S. 1071 (2015) (No. 14-410), 2015 WL 2457656 [hereinafter U.S. Amicus Brief] (giving more attention to the merger doctrine rationale).

138. *Borland*, 49 F.3d at 818.

139. *Id.*

140. *Id.*

141. 17 U.S.C. § 102(b).

142. *Borland*, 49 F.3d at 817.

143. *Id.* at 820 (Boudin, J., concurring).

144. *Id.* at 821. Courts should accord more weight to the *Borland* decision, albeit on Judge Boudin's rationale, in the aftermath of the Supreme Court's *Google* decision, which strongly endorsed the view that copyright's purposes are best served by allowing developers of computer programs to take advantage of having learned specific command terms and building programs with them. Google LLC v. Oracle Am., Inc., 141 S. Ct. 1183, 1203–04 (2021).

take advantage of a new advance, and to reward Borland for making a better product."[145] He agreed with the majority that Borland should prevail but was agnostic about whether holding the command hierarchy uncopyrightable was a better approach than articulating a privilege akin to fair use.[146]

In its amicus curiae brief to the First Circuit in support of Borland's appeal, the American Committee for Interoperable Systems (ACIS), of which Sun Microsystems was a founding member, noted the dual role of the Lotus 1-2-3 command hierarchy, serving both as a user interface through which users could interact with the program to invoke specific functions and as a program-to-program interface when enabling the execution of user-created macros.[147] ACIS pointed out that Lotus 1-2-3 users had spent considerable time and resources constructing libraries of macros in Lotus 1-2-3. It argued that Borland should be able to translate those macros to enable users to port the macros to its alternative platform.[148] ACIS characterized the set of rules or commands that enables two programs to exchange information and interoperate effectively—that is, its interface—as an unprotectable system or method of operation under § 102(b).[149] Allowing second comers to reimplement interfaces to enable compatibility free from copyright liability fosters competition and ongoing innovation in a manner consistent with copyright law's constitutional purposes.[150] We think ACIS offered the more persuasive rationale for ruling in Borland's favor.[151]

D. *Summary: Different Roads to the Same Destination*

Between 1992 and 2014, appellate courts had reached a strong consensus that achieving program compatibility was desirable and that copyright did not give programmers the ability to prevent others from reusing parts of another's program when necessary to enable compatibility.[152] There

145. *Borland*, 49 F.3d at 821 (Boudin, J., concurring).

146. *Id.* at 821–22.

147. ACIS-CCIA Borland Amicus, *supra* note 130, at 2.

148. *Id.* at 6.

149. *Id.* at 11. *See also* Taylor Instrument Cos. v. Fawley-Brost Co., 139 F.2d 98, 99–100 (7th Cir. 1943) (rejecting Taylor's claim of copyright aimed at stopping a competitor from making temperature recording charts that were compatible with Taylor's recording devices under Baker v. Selden, 101 U.S. 99 (1879)); Brown Instrument Co. v. Warner, 161 F.2d 910, 911 (D.C. Cir. 1947) (upholding Register's refusal to register Brown's recording charts, relying on *Baker* and *Taylor*).

150. ACIS-CCIA Borland Amicus, *supra* note 130, at 14–15.

151. The focus on the system and method of operation language in § 102(b) has been picked up in other cases outside the software context involving technical drawings. *See, e.g.*, RJ Control Consultants, Inc. v. Multiject, LLC, 981 F.3d 446, 454–55 (6th Cir. 2020) (copying of a copyrighted work held permissible where "the drawings were copied *to reproduce the control system contained therein*" (emphasis in original)).

152. *See, e.g.*, Lemley, *supra* note 82, at 12–17 (discussing the convergence by federal courts on the *Altai* filtration approach between 1992 and 1994); Pamela Samuelson & Suzanne Scotchmer,

was less agreement on the legal doctrine that compelled this result; courts variously relied on the *Altai* AFC test, the *scenes a faire* doctrine, the merger doctrine, § 102(b)'s exclusion for systems and methods of operation, and the fair use doctrine. We agree with the Ninth Circuit in *Accolade* and with the ACIS brief in *Borland* that the § 102(b) exclusion of procedures, systems, and methods is the most persuasive basis for excluding program interfaces that facilitate compatibility from the scope of copyright protection for computer programs,[153] although merger may be the most appropriate doctrine in cases such as *Lexmark* where exact copying of code, not just reimplementation of an interface, is at stake.

Regardless of which doctrinal hook courts decided to invoke, none had held that copyright law gave owners the power to prevent others from independently creating a compatible program. This does not mean computer programs get no copyright protection; they do. But that protection does not extend to APIs and other components necessary for compatibility.

Reinforcing the judicial consensus on this point, Congress endorsed program interoperability as part of the Digital Millennium Copyright Act in 1998.[154] While that Act made it illegal to circumvent a technical protection measure that controlled access to a copyrighted work, Congress was careful not to prohibit circumvention "for the sole purpose of identifying and analyzing those elements of the program that are necessary to achieve interoperability of an independently created computer program with other programs."[155] Only the Third Circuit remained an outlier, bound by its *Whelan* and *Franklin* precedents and unwilling to accept interoperability as a justification for copying or a limitation on copyright.[156] And there the law stayed until 2014, when the Federal Circuit first took up *Oracle III*.

The Law and Economics of Reverse Engineering, 111 YALE L.J. 1575, 1621–26 (2002) (examining the welfare effects of reverse engineering to achieve interoperability).

153. Systems and the rules for implementing them have long been held unprotectable by copyright law. *See, e.g.*, Pamela Samuelson, *Why Copyright Law Excludes Systems and Processes from the Scope of Its Protection*, 85 TEXAS L. REV. 1921, 1928–44 (2007) (discussing the system exception after *Baker*). *See also* Pamela Samuelson & Catherine Crump, *Why 72 Intellectual Property Scholars Support Google's Copyrightability Analysis in the* Oracle *Case*, 36 BERKELEY TECH. L.J. (forthcoming 2021), https://lawcat.berkeley.edu/record/1202388?ln=en [https://perma.cc/GN97-Z9KM] (explaining why program interfaces should be unprotectable by copyright law under *Baker* and § 102(b)).

154. Digital Millennium Copyright Act, Pub. L. No. 105-304, 112 Stat. 2860, 2866–67 (1998).

155. 17 U.S.C. § 1201(f)(1). Similar language also appears in the European Software Directive. Directive 2009/24/EC, of the European Parliament and of Council of 23 April 2009 on the Legal Protection of Computer Programs, paras. 10–11, 15, art. 1(2), 2009 O.J. (L 111) 16, 17–18.

156. *See* Dun & Bradstreet Software Servs., Inc. v. Grace Consulting, Inc., 307 F.3d 197 (3d Cir. 2002) (rejecting defendant's interoperability arguments). The facts of that case were decidedly less favorable to the claim of compatibility; the defendant used the plaintiff's code under license and apparently did some outright copying of plaintiff's code and misappropriation of trade secrets in an effort to divert plaintiff's customers while nominally working with the plaintiff. *Id.* at 219.

II. The Long Saga of *Google v. Oracle*

The Federal Circuit starkly deviated from two decades of consensus decisions when ruling that Java API declarations at issue in *Oracle III* were copyrightable. While the Supreme Court's *Google* decision reversed the Federal Circuit on other grounds, and therefore did not reach that question, we perceive in the Court's opinion support for our conviction that the consensus view, not the Federal Circuit's detour, is the right approach. Subpart A summarizes the district court and Federal Circuit decisions. Subpart B discusses Google's fate before the Supreme Court. Subpart C considers numerous statements in the Supreme Court's *Google* decision about the Java API and declarations that cast doubt on their copyrightability, which resonate with Google's merger and § 102(b) arguments. There are, moreover, significant similarities in the *Google* opinion's characterizations of programs and implications of these characterizations for copyright scope as appear in the pro-compatibility cases. Subpart D speculates about why the Court had so little to say about the role of software interfaces in enabling compatibility as compared with the uncopyrightability decisions discussed in Part I.

A. The District Court Got It Right, but the Federal Circuit Reversed Course

In the 1990s, Sun Microsystems developed the Java Programming Language as an interoperable programming environment in which programmers could "Write Once, Run Anywhere." Sun made most of its Java implementations available without charge, enabling Java to become a de facto standard.[157] Millions of software developers learned the Java language, as well as many of the standardized Java Application Programming Interface (API) packages, and used them to write applications for desktop and laptop computers and other devices.

Google adopted the widely known Java programming language when it designed its open smartphone platform for Android. It used thirty-seven of the 166 Java SE API packages to enable programmers familiar with Java to more easily develop Android apps. Google decided not to adopt Java API declarations from the other 129 packages because they were tailored for desktops and laptops and were not appropriate for the innovative smartphone platform that Android would become.

But the Third Circuit's rejection of the consensus approach was not expressly limited to those facts. *See id.* at 216 (dismissing industry practice and custom as justifications for the copying).

157. For a discussion of Java's de facto openness, see generally Mark A. Lemley & David McGowan, *Could Java Change Everything? The Competitive Propriety of a Proprietary Standard*, 43 ANTITRUST BULL. 715 (1998).

Google initially sought to obtain a compatibility license from Sun to use the thirty-seven API packages. But Sun insisted that Google implement all 166 API packages.[158] Google declined because it believed that handset makers would want a more permissive license in order to innovate new features, and the Android team concluded that including all of the API packages—many of which would have no applicability to smartphones—would undermine the speed, battery usage, and storage capacity of smartphones.[159]

Instead, Google wrote its own code to implement the thirty-seven selected API packages. It developed millions of lines of new code for the Android operating system—the implementation of the Java API and its own code to support new APIs relating to GPS, camera functions, user preferences, and other smartphone features. To enable programmers familiar with Java to work easily with the Android platform, Google copied roughly 11,500 lines from the Java SE program containing "declarations"—program elements necessary to achieve interoperability with the thirty-seven Java API packages and the titles Sun had given those calls.[160] The copied material constituted only 0.4% of the Java SE, and a far smaller fraction of Android.

After Oracle Corporation acquired Sun Microsystems in 2010, Oracle sued Google alleging infringement of the Java API.[161] At the trial court level, "Oracle's central claim . . . was that Google had replicated the structure, sequence and organization of the overall code for the 37 API packages."[162] Oracle claimed that Google had wrongfully copied the SSO of the Java API, that is, the structure embodied in the java.package.Class.method() hierarchy.

Relying on the judicial consensus that program interfaces that facilitate compatibility are unprotectable by copyright law, Google moved for summary judgment arguing that its reimplementation of parts of the Java API was not copyright infringement as a matter of law. Google invoked the *scenes a faire*, merger, § 102(b) method exclusion, and fair use doctrines as the basis for its motion.[163] The district court in *Oracle I*, like the Eleventh Circuit in *Bateman*, was unpersuaded by Google's per se uncopyrightability theory, and

158. *See* Peter S. Menell, *Rise of the API Copyright Dead?: An Updated Epitaph for Copyright Protection of Network and Functional Features of Computer Software*, 31 HARV. J.L. & TECH. 305, 364 & n.314 (2018) (describing Sun's opposition to the "forking" of its program that would result from Google licensing only thirty-seven of the 166 API packages).

159. *Id.* at 366 & n.322.

160. *Id.* at 405–06.

161. *Oracle I*, 810 F. Supp. 2d 1002, 1005 (N.D. Cal. 2011). Oracle's complaint included some patent claims, which is why Oracle's appeals went to the Federal Circuit.

162. *Oracle II*, 872 F. Supp. 2d 974, 975 (N.D. Cal. 2012). Thus, Oracle tried it as a nonliteral infringement case. *Id.*

163. *Oracle I*, 810 F. Supp. 2d at 1009–13.

so denied this motion.[164] Judge Alsup wanted to know more about the role of the interface elements at issue before assessing Oracle's copyright claim and Google's defenses.

Yet after a full trial and an extensive review of the software copyright case law, Judge Alsup concluded that the Java API elements that Google reimplemented in the Android platform were unprotectable methods or systems within the § 102(b) exclusions or unprotectable under the merger doctrine.[165] The fact that Sun and Oracle had both sought and obtained patents on some program interface designs reinforced his view that interfaces were more patent than copyright subject matter.[166]

Judge Alsup also took note that millions of lines of code had been written with the declarations from the thirty-seven Java API packages that "necessarily used the command structure of names at issue."[167] The owners of this code were the programmers who called upon these declarations, not Oracle. "In order for at least some of this code to run on Android, Google was required to provide the same java.package.Class.method() command system using the same names with the same 'taxonomy' and with the same functional specifications."[168] Google had taken "what was necessary to achieve a degree of interoperability—but no more, taking care . . . to provide its own implementations."[169] Judge Alsup considered this ruling to be consistent with Ninth Circuit precedents that had characterized the functional requirements for achieving compatibility with other programs to be unprotectable procedures under § 102(b).[170]

Given the appellate precedents from the First, Second, Sixth, Ninth, Tenth, and Eleventh Circuits characterizing program interfaces as uncopyrightable under the *scenes a faire*, merger, and § 102(b) doctrines, as well as fair use, it was somewhat surprising that the Federal Circuit overturned the trial court's ruling that the Java declarations were

164. *Id.* However, the court was persuaded that the names of the API functions were unprotectable. *Id.* at 1009–10.

165. *Oracle II*, 872 F. Supp. 2d at 997–1000. Judge Alsup's reliance on both grounds is consistent with other decisions that have invoked both § 102(b) method exclusions and the merger doctrine. *See, e.g.*, Hutchins v. Zoll Med. Corp., 492 F.3d 1377, 1383–85 (Fed. Cir. 2007) (analyzing the copyrightability of the process of cardiopulmonary resuscitation and standard instructions for performing that process under § 102(b) and merger); MiTek Holdings, Inc. v. Arce Eng'g Co., Inc., 89 F.3d 1548, 1556 n.19, 1557 n.20 (11th Cir. 1996) (analyzing copyright claim in a command tree structure under § 102(b) and merger).

166. *Oracle II*, 872 F. Supp. 2d at 996.

167. *Id.* at 1000.

168. *Id.* (emphasis omitted).

169. *Id.*

170. *Id.*

unprotectable by copyright law.[171] Like the Third Circuit in *Franklin* and *Whelan*, it regarded § 102(b) as merely a restatement of the idea/expression distinction.[172] On appeal, Oracle argued that Google had literally copied 11,500 lines of "declaring code," as well as the SSO of the Java packages.[173] This made the court perceive the *Oracle III* case as similar to the *Franklin* case, in that both involved literal copying of program code, and as similar to *Whelan* in that both involved nonliteral copying of program SSO.[174] The Federal Circuit regarded both the "declaring code" and the "SSO" as copyrightable as long as they were "original" within the meaning of § 102(a), as even Google admitted they were.[175]

Oracle also persuaded the Federal Circuit that Google's merger defense lacked merit because of the large number of choices that Sun/Oracle developers had when deciding how to structure and name the Java declarations.[176] The merger doctrine, in that court's view, was applicable only if the plaintiff had no alternative choices in structuring and naming the

171. *Oracle III*, 750 F.3d 1339, 1354 (Fed. Cir. 2014). It is worth noting that Sun Microsystems, the firm that developed the Java API, was an influential proponent of the legal position that program interfaces were and should be unprotectable by copyright law. Sun was a founding member of the ACIS that filed amicus curiae briefs in numerous high profile software copyright cases arguing, among other things, that program interfaces should be considered unprotectable under the merger and *scenes a faire* doctrines or the method exclusion embodied in § 102(b). A collection of the ACIS briefs in a series of software copyright compatibility cases can be found at COMPUT. & COMMC'NS INDUS. ASS'N, https://www.ccianet.org/interop/ [https://perma.cc/F4ZH-9FWY]. In one of these briefs, ACIS urged the Supreme Court to rule that program copyrights should not extend to APIs that enable the creation of interoperable programs:

> If the developer of one part of the environment can use copyright law to prevent other developers from writing programs that conform to the system of rules governing interaction with the environment—interface specifications, in computer parlance—the first developer could gain a patent-like monopoly over the system without ever subjecting it to the rigorous scrutiny of a patent examination.

ACIS-CCIA Borland Amicus, *supra* note 130, at 4–5 (Oracle was a member of ACIS at the time.).

Things have changed, as one of us worried they might. *See* Lemley & McGowan, *supra* note 157, at 751 (expressing concern regarding the implications of Sun obtaining IP rights in Java). Oracle's attempt to prevent interoperability in the *Google* case is "particularly ironic because Java was itself developed as a way of creating interoperability across platforms." Joseph Gratz & Mark A. Lemley, *Platforms and Interoperability in* Oracle v. Google, 31 HARV. J.L. & TECH. 603, 604–05 (2018). Since Sun did not release Java on an open-source basis, Lemley & McGowan worried that Sun would try to close the standard to others "to reap the benefits of widespread adoption." *Id.* at 604 n.6, 605 (citing Lemley & McGowan, *supra* note 157). And indeed, that is exactly what happened after Oracle bought Sun's assets. *Id.* at 605.

172. *Oracle III*, 750 F.3d at 1354–55.

173. Because Oracle had litigated its case against Google at the trial court level as a nonliteral copyright infringement case, Google tried to persuade the Federal Circuit that Oracle had waived the opportunity to raise literal infringement claims on appeal. The Federal Circuit did not find that argument persuasive. *Id.* at 1359.

174. *Id.* at 1356.

175. *Id.* at 1356–57, 1367.

176. *Id.* at 1359–62.

element allegedly infringed.[177] The Federal Circuit also rejected Google's *scenes a faire* argument in part because it concluded Google had not developed an adequate factual record on which this defense could be based and in part because *scenes a faire*, like merger, in that court's view, must be judged from the plaintiff's standpoint. If the names and structure of the Java API declarations were not commonplace at the time that Sun/Oracle developed them, the Federal Circuit concluded they could not be unprotectable under the *scenes a faire* doctrine.[178]

The Federal Circuit treated Google's interoperability arguments in a separate section of the opinion. It rejected these arguments as well, invoking the *Franklin* decision's dicta for the proposition that interoperability considerations are irrelevant to copyrightability.[179] The court also found Google's compatibility defense to be disingenuous as it had designed Android to be incompatible with Java platforms (that is, because Google did not use all 166 Java API packages in Android, many apps developed for the Android platform could not be executed on Java-compliant platforms and vice versa).[180] The court thought that compatibility considerations might be relevant to a fair use defense, but not to copyrightability.[181] Google relied on *Accolade* and *Connectix* in support of its argument that Ninth Circuit law precluded copyright protection for interfaces, but the Federal Circuit interpreted them as fair use cases only.[182]

Although the Federal Circuit was skeptical that Google could prevail on its fair use defense, it concluded that triable issues of fact precluded granting judgment to Oracle as a matter of law.[183] It remanded the case to the district court so the court could conduct a jury trial on fair use. After the jury rendered a verdict in favor of Google's fair use defense, Oracle sought to overturn this verdict, arguing that no reasonable jury could have decided Google's copying of the Java declarations was fair use. The district court rejected Oracle's argument in an opinion that articulated numerous fact issues on which

177. *Id.* at 1361. The court ignored that Sega's and Sony's interfaces constrained the design choices of Accolade and Connectix, a fact that was central to the Ninth Circuit rulings. *See supra* text accompanying notes 85–99. *See also* Samuelson, *supra* note 14, at 442–46 (discussing how the merger of idea and expression constrains design choices).

178. *Oracle III*, 750 F.3d at 1363–64. This conception of the *scenes a faire* doctrine is inconsistent with the *Altai* and *Lexmark* decisions.

179. *Id.* at 1368–72. The Federal Circuit cited approvingly to *Dun & Bradstreet Software Servs., Inc. v. Grace Consulting, Inc.*, 307 F.3d 197, 215 (3d Cir. 2002), for the proposition that only constraints on the plaintiff's choices are pertinent limits on copyrightability in its discussion of Google's interoperability defense. *Oracle III*, 750 F.3d at 1370.

180. *Oracle III*, 750 F.3d at 1371.

181. *Id.* at 1372. That promise proved illusory, however. In its later reversal of a jury's finding of fair use, the Federal Circuit said that compatibility had no bearing on the case. *Oracle V*, 886 F.3d 1179, 1206 n.11 (Fed. Cir. 2018).

182. *Oracle III*, 750 F.3d 1365–69.

183. *Id.* at 1373–77.

Google and Oracle had offered conflicting evidence, concluding that the jury must have been persuaded by Google's evidence over Oracle's.[184]

But on appeal to the Federal Circuit, Oracle once again prevailed. The Federal Circuit cited Supreme Court case law holding that fair use is a mixed question of law and fact.[185] It concluded that legal questions predominate in fair use cases, so it treated the jury's verdict as advisory except as to two "historical facts."[186] The fair use factors that it thought most strongly disfavored the jury's verdict were the first and the fourth: Google's use was commercial and nontransformative and this taking had harmed Oracle as to both actual and potential markets.[187] While characterizing the amount taken as qualitatively substantial, the court decided that this factor was at best neutral.[188] Although persuaded that the nature-of-the-work factor favored Google, the Federal Circuit concluded that Oracle was right that Google's copying was an unfair and infringing use of the Java API as a matter of law.[189]

B. *The Supreme Court Weighs In*

Google sought certiorari on both the copyrightability of APIs and fair use issues. Several software-industry amicus curiae briefs supported this petition, all but one of which urged the Court to grant the petition on the copyrightability issue.[190] Many amicus briefs argued that the Federal Circuit's *Oracle III* decision had upset settled expectations that software developers had had for decades that program interfaces were unprotectable

184. *Oracle IV*, No. C 10-03561 WHA, 2016 WL 5393938, at *15 (N.D. Cal. Sept. 27, 2016).

185. *Oracle V*, 886 F.3d at 1192–93 (citing Harper & Row, Publishers, Inc. v. Nation Enters., 471 U.S. 539, 560 (1985)).

186. *Id.* at 1193–96. It deferred to the jury verdict in respect of Oracle's claim that Google acted in bad faith and in respect of functional considerations affecting the nature of the work factor. *Id.* at 1202–05.

187. *Id.* at 1210. Factor 1 was discussed *id.* at 1196–1204, and Factor 4 was discussed *id.* at 1207–10.

188. *Id.* at 1205–07.

189. *Id.* at 1210. The court discussed the nature-of-the-work factor. *Id.* at 1204–05. However, the court thought this factor should have little significance in software cases; to do otherwise would negate Congress's intent to provide copyright protection to programs. *Id.* at 1205.

190. *See, e.g.*, Amicus Curiae Brief of Devs. All. in Support of Petitioner, Google LLC v. Oracle Am., Inc., 141 S. Ct. 1183 (2021) (No. 18-956), 2019 WL 913607 [hereinafter Developers Alliance Amicus Brief I] (asking the Court to grant Google's petition to provide much-needed clarity and resolve conflicting approaches of the lower courts); Brief of Amici Curiae Mozilla Corp. et al. in Support of Petitioner, *Google*, 141 S. Ct. 1183 (No. 18-956) [hereinafter Mozilla Amici Brief] (same); Brief of Amicus Curiae Red Hat, Inc. in Support of Petitioner, *Google*, 141 S. Ct. 1183 (No. 18-956) [hereinafter Red Hat Amicus Brief] (same). *See also* Brief of Microsoft Corp. as Amicus Curiae in Support of Petitioner, *Google*, 141 S. Ct. 1183 (No. 18-956), 2019 WL 932014 [hereinafter Microsoft Amicus Brief] (addressing only the fair use ruling).

by copyright law.[191] They relied upon *Baker v. Selden* and some of the software-interface cases for support of the proposition that interfaces were and should be uncopyrightable.[192] The briefs explained the importance of interoperability in today's interconnected world and for the existence of the open-source software industry.[193]

The Court granted certiorari on both issues.[194] Google's opening brief's discussion of the copyrightability issue focused more on the merger doctrine than the § 102(b) exclusion challenge.[195] Several amicus briefs filed in support of Google's appeal, most notably one filed by IBM Corp., focused only on the copyrightability issue.[196] While the amicus briefs filed in support of Oracle's defense of the Federal Circuit's rulings outnumbered those filed in support of Google's challenge,[197] Google had stronger software industry support than Oracle.[198] Some software industry briefs expressed concern about the harmful effects on competition and ongoing innovation in the

191. *See, e.g.*, Mozilla Amici Brief, *supra* note 190, at 3 (asserting that in the context of APIs, the *Oracle III* decision "upended decades of industry practice and the well-established expectations of [software] developers"); Red Hat Amicus Brief, *supra* note 190, at 18 (arguing that *Oracle III* disrupts "a decades-long understanding and expectation that programming interfaces are available for everyone to use in creating new products and services").

192. *See, e.g.*, Mozilla Amici Brief, *supra* note 190, at 3 (discussing how the software industry had flourished thanks to the lack of copyright protection for interfaces brought about by *Baker* and § 102(b)); Red Hat Amicus Brief, *supra* note 190, at 15–18 (describing how the judicial consensus that interfaces are not copyrightable emerged from *Baker* and the *Computer Associates* and *Borland* cases).

193. *See, e.g.*, Microsoft Amicus Brief, *supra* note 190, at 3–12 (highlighting the benefits of open source software and interoperability on innovation); Red Hat Amicus Brief, *supra* note 190, at 4 (explaining that the extension of copyright protection to interfaces would create significant barriers to innovation).

194. *Google*, 141 S. Ct. at 1195.

195. Brief for the Petitioner, *supra* note 5, at 21–34 (discussing merger); *cf. id.* at 17–18 (discussing § 102(b) exclusions). Google's brief devoted close to equal time to the fair use issue. *Id.* at 34–50.

196. *See* Brief for Int'l Bus. Mach. Corp. & Red Hat, Inc. as Amici Curiae Supporting Petitioner, *Google*, 141 S. Ct. 1183 (No. 18-956) [hereinafter IBM Amici Brief] (arguing that interfaces are excluded from copyright under 17 U.S.C. § 102(b)); Amicus Curiae Brief of Devs. All. in Support of Petitioner, *Google*, 141 S. Ct. 1183 (No. 18-956), 2020 WL 224319 [hereinafter Developers Alliance Amicus Brief II] (same); Brief of Amicus Curiae Engine Advoc. in Support of Petitioner, *Google*, 141 S. Ct. 1183 (No. 18-956), 2020 WL 242504 [hereinafter Engine Advocacy Amicus Brief] (same).

197. Thirty-one amicus briefs supported Oracle, nine of which were filed on behalf of non-software copyright industry groups (e.g., the Motion Picture Association) or individuals (e.g., former Register of Copyrights Ralph Oman). Twenty-seven amicus briefs supported Google. Two amicus briefs were filed in support of neither party.

198. Google-side amici included IBM Corp., Microsoft Corp., and the App Developers Alliance. The most prominent of Oracle-side software industry amici were Dolby Labs., Inc. and SAS Institute, Inc., the latter of which as we note below has a similar case to Oracle's pending.

software industry if the Court did not decide that program interfaces were unprotectable by copyright law.[199]

The Supreme Court reversed the Federal Circuit's holding that Google was liable for copyright infringement, albeit on the fair use ruling instead of the copyrightability ruling.[200] Although this Article's primary focus is on the copyrightability issue, and although its authors would have preferred for the Court to have ruled that program interfaces that enable compatibility are unprotectable by copyright law, the Court's ruling that Google's reimplementation of the Java API declarations was fair use as a matter of law was a major victory not only for Google, but for the many programmers and developers who supported Google's appeal.

The Court agreed with the Federal Circuit that the nature-of-the-work factor favored Google's fair use defense but regarded that factor as much more significant than had the Federal Circuit.[201] The Court noted that the declarations and implementing code "call for, and reflect, different kinds of capabilities" and embody "different kind[s] of creativity." It concluded that the declarations were farther from the core of copyright than the implementing code, so there was less reason to be concerned about the risk of undermining incentives to create software.[202] The Court took into account that Sun had wanted programmers to learn Java and that a great deal of the value of the declarations lay in investments that programmers had made in learning and building programs using the declarations.[203]

The Court regarded Google's use of the declarations as "transformative" because this use contributed to the creation of an innovative smartphone platform and further enabled programmers to be highly creative in developing apps to run on that platform.[204] Although acknowledging that

199. *See, e.g.*, Developers Alliance Amicus Brief II, *supra* note 196, at 18–20 ("If . . . the use and implementation of API declarations are subject to individual control, then these interfaces become a choke point for monopoly control.").

200. *Google*, 141 S. Ct. at 1190.

201. *Id.* at 1201–02. To signal how important it regarded the nature-of-the-work factor, the Court addressed it first. *See id.* (explaining that the Court viewed the declaring code at issue as "further than . . . most computer programs . . . from the core of copyright" and as such the Court was less concerned than the Federal Circuit that a finding of fair use would undermine general copyright protection for computer programs).

202. *Id.* at 1202.

203. *Id.* Scholars have previously noted the nature-of-the-work factor rarely affected the outcome of cases. *See* Joseph P. Liu, *Two-Factor Fair Use?*, 31 COLUM. J.L. & ARTS 571, 572 (2008) (discussing how courts have expressly deemphasized the nature-of-the-work factor); Clark D. Asay, Arielle Sloan & Dean Sobczak, *Is Transformative Use Eating the World?*, 61 B.C. L. REV. 905, 960–61 (2020) (discussing how the nature-of-the-work factor carries little weight in the fair use analysis). For a pre-*Google* argument to reinvigorate that factor, see Samuelson & Asay, *supra* note 16, at 561.

204. *Google*, 141 S. Ct. at 1203–04. Although the Court regarded Google's use as commercial, this was "not dispositive . . . in light of the inherently transformative role that the reimplementation played in the new Android system." *Id.* at 1204.

Google's use of 11,500 declarations might seem a substantial amount, the percentage of the Java API Google used was relatively small.[205] In addition, the Court concluded that the use was reasonable in light of Google's purpose in allowing programmers experienced with them to build new programs for Android.[206]

The Court strongly disagreed with the Federal Circuit on the harm-to-the-market factor.[207] The Court largely deferred to the jury's verdict, noting that it had heard evidence about Oracle's claims of harm and must not have been persuaded by either its actual or potential market harm claims.[208] Moreover, the Court thought that enforcing Oracle's copyright claim would interfere with, rather than foster, fulfillment of the constitutional purpose of copyright.[209] The Court concluded that Google's reimplementation of the Java declarations was fair use as a matter of law.[210]

C. Some Parts of the Google *Decision Cast Doubts on the Copyrightability of Interfaces*

While the *Google* decision stated that the Court "assume[d], but purely for argument's sake," that the Java API declarations were copyrightable,[211] nowhere did the Court express support for *Oracle III* or any of the Federal

205. *Id.* at 1204–05.

206. *Id.* at 1205–06.

207. *Id.* at 1206–08.

208. *Id.* at 1206–07. The Court agreed with the Federal Circuit and Oracle that fair use is more a legal than a factual issue. *Id.* at 1199–1200. However, unlike the Federal Circuit, the Court regarded whether a challenged use had caused market harm to be a factual issue for the jury. *Id.* at 1200.

209. *Id.* at 1208.

210. *Id.* at 1209 ("[W]here Google reimplemented a user interface, taking only what was needed to allow users to put their accrued talents to work in a new and transformative program, Google's copying of the Sun Java API was a fair use of that material as a matter of law.").

211. *Id.* at 1197. The Court proffered this reason for not reaching the copyrightability ruling: "Given the rapidly changing technological, economic, and business-related circumstances, we believe that we should not answer more than is necessary to resolve the parties' dispute." *Id.* We question the strength of this rationale given the overwhelming software industry support for overturning the Federal Circuit's copyrightability ruling. *See, e.g.*, IBM Amici Brief, *supra* note 196, at 31 (urging the Court to reverse the Federal Circuit's copyrightability ruling); Developers Alliance Amicus Brief II, *supra* note 196, at 31 (same); Brief of Amici Curiae the Comput. & Commc'n Indus. Ass'n & Internet Ass'n in Support of Petitioner at 31, *Google*, 141 S. Ct. 1183 (No. 18-956), 2020 WL 224320 [hereinafter CCIA Amici Brief] (same). Moreover, the Court had denied certiorari on Google's first petition back in 2015 based in part on the Solicitor General's recommendation that the Court wait until the case was over so it could address all the issues together. *See* U.S. Amicus Brief, *supra* note 137, at 22 (suggesting that "[a]t a minimum, this Court could better assess and clarify the relevance of those concerns to copyright-law analysis if the Court had before it all potentially relevant statutory arguments").

Circuit's doctrinal analyses of the copyrightability issue.[212] Only the two dissenters reached that issue, and they would have held the Java declarations copyrightable.[213]

While the Court did not reach the copyrightability issue in *Google*, finding fair use a sufficient ground for its decision, there is much in the Court's analysis and reasoning that supports the pre-Federal Circuit consensus that APIs are uncopyrightable.

The Supreme Court's *Google* decision came closest to saying that the Java API declarations were uncopyrightable when describing the relationship among three key elements: method calls, with which programmers identify tasks they want to have performed (and in which Oracle did not claim copyright); implementing code, which actually carries out the appropriate tasks (which is copyrightable but which Google did not copy); and declarations which provide the link (that is, interface) between the method calls and implementing code.[214] The Court stated that the declarations are "inextricably bound up with" method calls as well as with implementing code.[215] The Court used the phrase "inextricably bound up" four times in one paragraph in describing the relationship between the Java declarations and method calls, implementing code, the division of computing tasks, and the organization of tasks into "what we have called cabinets, drawers, and

212. The only citations to *Oracle III* were in the section that explained what an API is and in the section reviewing the case history. *Google*, 141 S. Ct. at 1191, 1194–95. However, there was one respect in which the Court's language was similar to the Federal Circuit's copyrightability decision: in its acceptance of the term "declaring code," *Oracle III*, 750 F.3d 1339, 1349 (Fed. Cir. 2014), instead of "declarations," the term used by the district court and computer scientists. *Oracle II*, 872 F.2d 974, 979 (N.D. Cal. 2012); Brief Amici Curiae of 83 Computer Scientists in Support of Petitioner at 3, *Google*, 141 S. Ct. 1183 (No. 18-956) [hereinafter 83 Computer Scientists Amici Brief]. It is worth noting that Google acceded to the term declaring code in its Supreme Court briefings. Brief for the Petitioner, *supra* note 5, at 39.

213. Justice Thomas's dissent agreed with Oracle that APIs were copyrightable, *Google*, 141 S. Ct. at 1211, but his opinion garnered only two votes. Yet it was not just Justices Thomas and Alito who were unsure about how to distinguish declarations and implementing code as a copyrightability matter given that both fell within the definition of computer program in 17 U.S.C. § 101. *See* Transcript of Oral Argument at 14–15, *Google*, 141 S. Ct. 1183 (No. 18-956) (Breyer, J. asking Mr. Goldstein to distinguish between them). The dissent takes the (incorrect) position that "declaring code" must be understood to fall within the statutory definition of computer program in the copyright statute, 17 U.S.C. § 101, just as does implementing code, from which it concludes that if software is entitled to some statutory protection, then copyright must extend to the functional aspects of software. *Google*, 141 S. Ct. at 1212–13 (Thomas, J., dissenting). It views this as sufficient to eliminate both the methods of operation and merger doctrines. *Id.* at 1213–14. The dissent's approach finds no support in the statute or in any prior case law. Even those cases that found copyright infringement in software would not read the copyright statute as broadly as the dissent does. Further, the dissent makes no reference to the fact that the code in question related to APIs, and draws no distinction at all between APIs and other forms of code. For criticism of the dissent's approach to copyrightability, see JONATHAN BAND, INTERFACES ON TRIAL 3.0: GOOGLE V. ORACLE AMERICA AND BEYOND 126–27 (2021).

214. *Google*, 141 S. Ct. at 1201.

215. *Id.*

files."[216] Of these four elements with which declarations are linked, the Court said only implementing code was copyrightable.[217] Use of the declarations, moreover, is "inherently bound together with uncopyrightable ideas (general task division and organization) and new creative expression (Android's implementing code)."[218] These statements were supportive of Google's merger arguments under which expression that is "inextricably bound up with" ideas or function is not copyrightable.[219]

There is also good reason to think that the majority approached the Java API as an uncopyrightable system or method of operation. The Court spoke of the Java API's overall organization as a "system" and of the declarations as part of a "task calling system."[220] The Court repeatedly expressed concern about programmers having to learn a new "system" to call up the same tasks instead of being able to use that with which they were already familiar.[221] The Court characterized declarations as "inextricably bound together with a general system, the division of computing tasks, that no one claims is a proper subject of copyright."[222] Google copied the declarations "not because of their creativity, their beauty, or even (in a sense) because of their purpose," but "because programmers had already learned to work with the Sun Java API's system."[223] Each of these statements echoes the First Circuit's treatment of the same issues in its *Borland* decision.[224] These statements provide some support for Google's § 102(b) system arguments.[225]

Further, all of the non-computer analogies the Court used to explain the Java API were to things copyright doesn't protect. The *Google* decision likened the Java API declaration structure to file cabinets, drawers, and files and more generally analogized the declarations to the Dewey Decimal

216. *Id.*

217. *Id.*

218. *Id.* at 1202. *See also id.* at 1205 ("[T]he Sun Java API is inseparably bound to those task-implementing lines," for "[i]ts purpose is to call them up.").

219. *See* discussion of *Lexmark supra* notes 108–122. *See also* Samuelson, *supra* note 66, at 1267–84 (discussing the origins, role, and function of the merger doctrine in software cases). The word "merger" appears only once in the *Google* opinions, when noting that Google had invoked this doctrine as a defense. *Google*, 141 S. Ct. at 1213 (Thomas, J., dissenting).

220. *Google*, 141 S. Ct at 1192–93.

221. *Id.* at 1193–94, 1205.

222. *Id.* at 1201.

223. *Id.* at 1205.

224. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815–19 (1st Cir. 1995).

225. Justice Breyer's opinion mentioned 17 U.S.C. § 102(b) twice, but only in background sections. *Google*, 141 S. Ct. at 1194, 1196. After quoting from the text of § 102(b), it noted that this provision was relevant to the copyright–patent distinction. *Id.* at 1196. Google had made much of that distinction in its briefs and during oral argument. *See, e.g.*, Brief for the Petitioner, *supra* note 5, at 17–18, 23 ("[C]opyright protection in a work extends only to expression, not to the idea that the expression conveys."); Transcript of Oral Argument, *supra* note 213, at 3–4, 7 (arguing that "Oracle has a copyright to the computer code in Java SE but not a patent"). This argument got no traction with the Justices. *Id.* at 7.

System, gas pedals that enable cars to move faster, QWERTY keyboards, travel-guide categories, and most strangely, human keystrokes that might direct robots to fetch recipes and deliver them to cooks.[226] Copyright law would not extend protection to any of these intellectual creations (except for the software in the robot). Thus, these analogies offer further support for Google's uncopyrightability arguments.

The *Google* decision also made numerous statements that resonate with those found in appellate court decisions in which compatibility defenses succeeded. For instance, the Court likened the Java API to the user interface menu commands held unprotectable in *Borland*.[227] As in *Altai* and *Accolade*, the Court in *Google* characterized computer programs as functional works that enjoy a thinner scope of copyright protection than artistic or fictional works.[228] As in *Altai*, the Court acknowledged that it is difficult to apply copyright doctrines to computer programs because of their intermixture of functionality and expressiveness.[229] As in *Accolade* and *Lexmark*, the Court perceived the need for a "context-based check that can help to keep a copyright monopoly within its lawful bounds."[230] The Court's warnings about lock-in effects and risks that copyright might stifle innovation resonate with similar concerns in the *Altai*, *Accolade*, and *Borland* decisions.[231] The *Google* decision's emphasis on the importance of enabling programmers to reimplement interfaces is more consistent with *Altai* and its progeny than with *Whelan/Oracle III*.[232] The Court notably cited approvingly to *Borland*, *Accolade*, *Connectix*, and *Lexmark*, each of which had characterized program

226. *Google*, 141 S. Ct. at 1192–93. During oral argument, the Justices tested out several other analogies: the headings of briefs, math class proofs, the arrangement of food items on restaurant menu, telephone switchboards, periodic tables, and phyla of living things. Transcript of Oral Argument, *supra* note 213, at 5, 27, 40–41, 47, 57.

227. *Google*, 141 S. Ct. at 1201.

228. *Id.* at 1198.

229. *Id.* at 1198, 1208. The Court quoted from Judge Boudin's concurrence in *Borland* in observing that "applying copyright law to computer programs is like assembling a jigsaw puzzle whose pieces do not quite fit." *Id.* at 1198 (quoting *Borland*, 49 F.3d at 820 (Boudin, J., concurring)).

230. *Id.* Indeed, the Court cited *Lexmark* and *Accolade* on this point. *Id.* at 1198–99.

231. *Id.* at 1198, 1208.

232. It is telling, however, that the *Google* decision cited *Baker* only once, and then only in passing, as a cf. to a statement about "decisions about what counts as a task" not being copyrightable, although "one might argue about decisions as to how to label and organize such tasks (e.g., the decision to name a certain task 'max' or to place it in a class called 'Math.'[])." *Id.* at 1201. This suggests that the Justices were unable to reach consensus on the implications of *Baker* for the copyrightability of the Java API declarations. Many of the Google-side briefs, from industry amici, public interest advocates, as well as academics, relied on the logic of *Baker* and its codification in § 102(b) in support of their arguments that the Java API declarations were uncopyrightable. *See, e.g.*, IBM Amici Brief, *supra* note 196; Engine Advocacy Amicus Brief, *supra* note 196; Brief of Amicus Curiae Elec. Frontier Found. in Support of Petitioner, 141 S. Ct. 1183 (No. 18-956); Brief of 72 Intellectual Prop. Scholars as Amici Curiae in Support of Petitioner, 141 S. Ct. 1183 (No. 18-956). The *Google* decision also cited § 102(b) only in background sections of the opinion. *Google*, 141 S. Ct. at 1194, 1196.

interfaces needed for compatibility as unprotectable by copyright law.[233] Moreover, it never cited to *Whelan* or *Franklin* or embraced the protectability of program SSO or expressed hostility to competitors' desires to achieve interoperability.

D. *Why Did the Court Have So Little to Say About Compatibility?*

The terms "compatible/compatibility" and "interoperable/ interoperability" were rarely mentioned in the *Google* decision.[234] The term "interface" appeared occasionally, but mostly in characterizing declarations as providing a "user interface" between programmers and the machines on which their programs will run.[235] Yet, it is fair to interpret the *Google* decision as having implicitly recognized a somewhat different kind of interoperability than consensus cases had addressed: interoperability of humans with software, such that programmers could continue to use declarations that they already knew when writing apps for the Android platform as well as for Java-compliant platforms.

The rarity of references to "compatible/compatibility" and "interoperable/interoperability" in *Google* may seem surprising given how widely invoked, indeed almost ubiquitous, those terms were in the software industry amicus briefs filed in support of Google.[236] We will turn in Part III to what we think the Court should have said about compatibility and why, but before doing so, it is worth considering why these terms had so little traction with the Court in the *Google* case.

The most obvious explanation is that the interoperability considerations expressed in the Google-side briefs did not mesh well with the predominant narrative of Justice Breyer's fair use opinion: that Google's reimplementation of the Java API declarations was fair use because this allowed programmers

233. *Google* 141 S. Ct. at 1198–99. At least three of the amicus briefs from which the Court quoted as supporting Google's appeal were briefs that argued that APIs should be unprotected by copyright law. *Id.* at 1203–04 (first citing Brief of the R St. Inst. et al., *Google*, 141 S. Ct. 1183 (No. 18-956); then citing 83 Computer Scientists Amici Brief, *supra* note 212; and then citing Brief for the Am. Antitrust Inst. as Amicus Curiae in Support of Petitioner, *Google*, 141 S. Ct. 1183 (No. 18-856)).

234. *Id.* at 1190 (mentioning "interoperable" in relation to the development of Java to enable software to run on multiple platforms), 1191 (noting that "Sun's interoperability policy would have undermined" Google's business model), 1198 (mentioning "compatibility" in a parenthetical citing to *Lexmark*), 1205 (referring to Google's "Java-compatibility objective" in reusing Java declarations). The Court obliquely noted that "[t]he jury heard that shared interfaces are necessary for different programs to speak to each other." *Id.* at 1203.

235. *Id.* at 1192, 1201–04, 1208–09.

236. *See, e.g.*, Developers Alliance Amicus Brief II, *supra* note 196 *passim* (using "interoperable/interoperability" repeatedly); Microsoft Amicus Brief, *supra* note 190 *passim* (using "compatible/compatibility" and "interoperable/interoperability" repeatedly). *See also* 83 Computer Scientists Amicus Brief, *supra* note 212 *passim* (using "compatible/compatibility" and "interoperable/interoperability" repeatedly).

who had invested in learning the declarations to continue using them as building blocks with which to create new programs that could run on the creative Android smartphone platform.[237]

The Court's view about Google's motivations in reusing the declarations contrasts sharply with the Federal Circuit's. The Federal Circuit viewed Google as having copied the declarations "to capitalize on the preexisting community of programmers who were accustomed to using the Java API packages,"[238] as though by trying to attract programmers to write for Android, Google was engaged in unfair competition with Oracle and was free riding on Oracle's innovations. The Supreme Court clearly rejected that view. One reading of the *Google* decision is that interoperability per se didn't matter to the Court's opinion; Google's use was permissible whether or not the programs were compatible so long as it allowed third party programmers to continue to use programming languages they knew.

Another possible explanation for the Court's lack of attention to compatibility issues is that the Court may not have conceived of *Google v. Oracle* as a compatibility case in the strict sense.[239] *Altai*, *Accolade*, *Bateman*, *Lexmark*, and *Borland* were "truer" compatibility cases because the defendants in those cases really did need to reimplement other firms' interfaces in order for their programs to successfully interoperate with other programs.[240] *Google* was not such a case.

Google can more accurately be said to be about enabling cross-platform consistency than about program-to-program interoperability.[241] The Android software, after all, was designed to run on a very different type of computing device—smartphones—than the laptop and desktop computers for which Java SE was originally developed. Google did not reimplement in Android most parts of the Java API because the 129 packages it did not reimplement enabled functionalities for laptops and desktops that smartphones did not

237. *Google*, 141 S. Ct. at 1202–03 (discussing how the value of the declarations derives in significant part from the investments programmers made in learning them and using them to create new programs).

238. *Oracle III*, 750 F.3d 1339, 1372 (Fed. Cir. 2014).

239. During oral argument, Justice Sotomayor asked Google's counsel about his definition of interoperability. She seemed to think that his definition was one-directional. She asked if others could copy Google's declarations for an alternative platform. Transcript of Oral Argument, *supra* note 213, at 23. Mr. Goldstein indicated that Google didn't use all of the declarations because smartphones are a different type of platform. *Id.* at 23–24.

240. During oral argument, Justice Sotomayor noted that since 1992, *Altai* and cases from at least three other circuits had held that APIs were not copyrightable, thus shaping industry expectations about the unprotectability of interfaces. Why, she asked, should the Court upset settled expectations? Transcript of Oral Argument, *supra* note 213, at 52–54. Justice Kagan also mentioned *Altai* as having articulated a test for separating functionality and expression in programs. *Id.* at 26.

241. *See* Ord. Denying Rule 50 Motions, *Oracle IV*, 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA), 2016 WL 3181206, at *6 ("Google copied only so much declaring code as was necessary to maintain inter-system consistency among Java users.").

need (e.g., enabling mouse functionalities).[242] The thirty-seven Java API packages that Google reimplemented were those that enabled tasks that all types of computing devices must be able to perform (e.g., comparing two numbers to determine which was larger). Because of the many significant differences between smartphones and laptops/desktops as computing platforms, Google had to develop many new declarations to enable performance of smartphone-specific tasks.

As a result of these differences, some programs initially developed for Java SE-compliant platforms could not be executed on the Android platform, and some programs developed for Android would not run on Java SE-compliant devices. Although some programs could be executed on both Android and Java SE-compliant platforms without adaptation, many programs first developed for one of these platforms would not work on the other without some adaptation.

In this respect, the *Google* case is distinguishable from other compatibility decisions upholding uncopyrightability defenses. This may have been another reason why the Court chose not to discuss *Altai* and its progeny and why the Court hardly ever used the terms "compatible/compatibility" or "interoperable/interoperability" when discussing the *Google* dispute. It certainly mattered to the Federal Circuit, which dismissed all compatibility arguments out of hand on the grounds that Android was not fully interoperable with Java.[243] As we discuss in the next Part, however, compatibility need not be perfect to be valuable.

III. APIs and the Importance of Interoperability

The law before *Google v. Oracle* was clear and well-settled: software developers could not use copyright law to prevent others from copying APIs to achieve compatibility. The Federal Circuit's decision departed from those precedents in holding that Sun/Oracle's API declarations were copyrightable. While the Supreme Court did not find it necessary to address the 2014 decision, both the Court's rationale and its language suggest that—had fair use not been at issue in the case—it would follow the overwhelming consensus of the regional circuits, not the Federal Circuit's reinterpretation of copyright law. Indeed, Justice Thomas's dissent suggests that the majority opinion is tantamount to an uncopyrightability ruling.[244]

242. *Google*, 141 S. Ct. at 1203.

243. *Oracle III*, 750 F.3d at 1368–71.

244. *Google*, 141 S. Ct. at 1214 (Thomas, J., dissenting) ("The result of this distorting analysis is an opinion that makes it difficult to imagine any circumstance in which declaring code will remain protected by copyright.").

Nonetheless, and even though the Federal Circuit is supposed to defer to the regional circuits in interpreting copyright law,[245] some software developers are likely to engage in forum shopping so that their appeals will go to the Federal Circuit and to argue that it should follow its 2014 *Oracle III* decision rather than Supreme Court or regional circuit precedent. Indeed, they have already done so in one case pending before the Federal Circuit right now.[246]

In this Part, we argue first that defendants in software copyright cases should not have to rely only on fair use when sued for copyright infringement based on their reimplementations of interfaces that facilitate compatibility, and then that courts should continue to protect interoperability by denying copyright protection to APIs. The Federal Circuit does not have authority to establish its own copyright precedents. It is supposed to follow the law of the regional circuits, so it should acknowledge and defer to those decisions.[247]

A. Why Fair Use Is Not Enough

The Court's strong endorsement of Google's fair use defense may deter some software developers from bringing lawsuits to challenge firms that reimplement their interfaces. However, reimplementers who operate in the same or a closely proximate market as the interface's developer may consider *Google* to be distinguishable. After all, the Court made much of Android's being in a different market segment from Java SE when considering the important market harm factor.[248] It also remains to be seen how courts will construe the *Google* decision when second comers have reimplemented other interface elements of programs such as command structures, command titles, and input/output formats.

There are, however, other reasons not to place full reliance on fair use. Because it is a fact-and-case-specific, multi-factor test, it is hard to predict

245. *See, e.g.,* Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832, 837 (Fed. Cir. 1992) ("To resolve issues of copyright law, this court applies the law as interpreted by the regional circuits.").

246. SAS Inst., Inc. v. World Programming Ltd., 496 F. Supp. 3d 1019 (E.D. Tex. 2020) (dismissing SAS's copyright claim for failure to identify protectable and unprotectable elements of its software), *appeal docketed*, No. 2021-1542 (Fed. Cir. Jan. 13, 2021) (Westlaw).

247. *See, e.g., Oracle III*, 750 F.3d at 1353 (noting that "[w]hen the questions on appeal involve law and precedent on subjects not exclusively assigned to the Federal Circuit," the Federal Circuit applies the regional circuit's law (quoting *Atari Games*, 897 F.2d at 1575)). Yet, the Federal Circuit failed to acknowledge that the Ninth Circuit in *Accolade* and *Connectix* had characterized program interfaces needed for compatibility as unprotectable procedures under 17 U.S.C. § 102(b). *Id.* at 1365–68. Nor did the Federal Circuit attempt to distinguish pro-compatibility cases such as *Altai* and *Bateman* from the interface elements at issue in the *Oracle III* dispute. The Federal Circuit's interpretation of copyright law, as applied to interfaces, called into question the holdings of all of the pro-compatibility cases. That may explain why so many key players in the software industry supported Google's appeal of the Federal Circuit's copyrightability ruling.

248. *Google*, 141 S. Ct. at 1206–07.

the outcome of fair use defenses early. Indeed, Larry Lessig once described fair use as nothing more than "the right to hire a lawyer."[249] While the Supreme Court's holding that many core issues of fair use can be resolved without the need for a jury is helpful in reducing uncertainty, fair use still depends quite heavily on how judges view subjective issues like the transformative nature of the work. As the Second Circuit's remarkable decision in *Andy Warhol Foundation v. Goldsmith*[250] reminds us, courts can upend even the seemingly most obvious and simple assumptions about fair use.[251] If an appellate court can conclude that there is nothing transformative or creative about Andy Warhol's paintings, it's hard to predict with any certainty how a computer program will fare.

Consider also who bears the ultimate burden of persuasion when defendants raise uncopyrightability and fair use as grounds for questioning copyright claims involving reuses of interfaces that facilitate compatibility. Plaintiffs always bear the burden of proving that what the defendants copied from their programs was copyright-protectable expression.[252] Courts generally characterize fair use as an affirmative defense on which defendants, not plaintiffs, bear the burden of persuasion (which is a view with which we disagree).[253]

Even beyond the problem of subjective judgment and proof burdens, fair use still depends on some case-specific questions of fact, particularly regarding market effect. The Supreme Court deferred to the jury's implicit finding that Oracle didn't suffer market harm because neither it nor Sun had a realistic prospect of entering the market for smartphone operating systems. But that leaves open the possibility that a jury in a different case with different facts might reach a different conclusion about market harm.

There is a particular risk that the *Google* fair use decision might not extend to cases where the parties, unlike Google and Oracle, are direct competitors. The pro-interoperability caselaw has protected companies who

249. LAWRENCE LESSIG, FREE CULTURE: THE NATURE AND FUTURE OF CREATIVITY 187 (2004).

250. 992 F.3d 99 (2d Cir. 2021), *withdrawn and superseded on reh'g*, No. 19-2420-cv, 2021 WL 3742835 (2d Cir. Aug. 24, 2021). The amended opinion drops some of the statements most obviously in conflict with *Google* but does not change the shocking result or basic analysis.

251. *See id.* (reversing a trial court's summary judgment ruling that Warhol had made a transformative fair use of a Goldsmith photograph of singer Prince). To be clear, we believe the Second Circuit got this case very wrong, and we are hopeful that the court will correct its error in view of *Google*'s statements about transformative use. *See* Brief of Amici Curiae 60 Intell. Prop. Scholars in Support of Petition for Panel Rehearing and Rehearing En Banc, *Warhol*, 992 F.3d 99 (2d Cir. 2021) (No. 19-2420-cv) (supporting the foundation's petition for rehearing and arguing that the Second Circuit panel's decision is inconsistent with the fair use analysis in *Google*).

252. *See, e.g.*, Feist Pub., Inc. v. Rural Telephone Service Co., 499 U.S. 340, 361 (1991) (noting that plaintiffs must prove ownership or a valid copyright and copying of constituent elements of the work that are original).

253. *See supra* note 17.

copy for purposes of compatibility even when they do so to compete directly with the copyright owner.[254] But the Supreme Court's decision might not be read to extend so far. And copyright owners can and will take advantage of any perceived uncertainty to try to shut down compatibility.

Fair use is, moreover, less desirable from the software industry's standpoint than uncopyrightability in that defendants will have to spend valuable resources in litigating fair use defenses, which are rarely resolved through motions to dismiss. And the most innovative advances in the software industry often come from smaller developers who can least afford to litigate fair use.

B. *The Value of Interoperability for Consumers and Creators*[255]

Computer programmers and software companies have long relied on the pro-compatibility settled case law because they depend on their freedom to reimplement APIs as part of the invisible infrastructure of modern computing.[256] Computers and the internet "work" because different programs and devices can communicate with each other. Interoperability makes that possible.[257] Interoperability is the reason you can read a website regardless of what internet browser you use.[258] It's the reason you can read documents on a PC even though someone wrote them on a Mac.[259] It's the reason messages can pass from phone to computer to tablet and back again.[260] And it's the reason you can use search engines to find data. The modern networked

254. Sony Comput. Ent., Inc. v. Connectix Corp, 203 F.3d 596, 607 (9th Cir. 2000).

255. Portions of this subpart are adapted from Gratz & Lemley, *supra* note 171.

256. *See, e.g.*, Brief of Comput. Scientists as Amici Curiae in Support of Defendant-Appellee at 1–3, *Oracle V*, 886 F.3d 1179 (Fed. Cir. 2018) (Nos. 2017-1118, 2017-1202) (brief of seventy-six widely recognized computer scientists) (arguing that the "software industry has long relied on and benefitted from the open nature of application programming interfaces").

257. *Id.* at 10–14.

258. *Cross-Browser Compatibility Tutorial: Use JS for Cross-Browser Compatibility*, YOUTUBE (Sept. 30, 2015), https://www.youtube.com/watch?v=FGAV4UMvedk [https://perma.cc/N836-7AQP]; Richard Cornford, *Browser Detection (and What to Do Instead)*, JIBBERING, http://jibbering.com/faq/notes/detect-browser/ [https://perma.cc/NK7X-QRDH]; *What Is JavaScript?*, MDN WEB DOCS, https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript [https://perma.cc/JA2V-LF9T].

259. Gina Trapani, *The Complete Guide to Mac/Windows Interoperability*, LIFEHACKER (Oct. 19, 2007), https://lifehacker.com/311618/the-complete-guide-to-macwindows-interoperability [https://perma.cc/QC9A-XPLU]; *see also* Erik Eckel, *Mac vs. Windows Incompatibility Achieves Irrelevance*, TECHREPUBLIC (Feb. 16, 2015), http://www.techrepublic.com/article/mac-vs-windows-incompatibility-achieves-irrelevance/ [https://perma.cc/6TLG-KQ5H] (explaining that Microsoft Office compatibility issues have been eliminated).

260. *See generally What Is Cross-Platform Software?*, BOBOLOGY, https://www.bobology.com/public/What-is-CrossPlatform-Software.cfm [https://perma.cc/M3FG-CQYE]; *see also* Daniel Nations, *4 Ways to Develop for iOS, Android, Windows, and Mac at the Same Time*, THOUGHTCO. (Sept. 24, 2020), https://www.lifewire.com/develop-for-ios-android-windows-mac-1994294 [https://perma.cc/Z75J-TDXH] (explaining how to simultaneously develop applications across different platforms).

world would be impossible without interoperability. And interoperability wouldn't be possible without freedom to reimplement APIs. They provide open, documented sets of instructions that third parties can use to ensure that information can successfully pass back and forth between programs.

Some interoperability is due to the voluntary development of open standards. Many companies may want to allow others to create compatible programs.[261] For example, the JavaScript standard—which, despite the name, bears no relationship to Java—was published as an open standard.[262]

But court decisions refusing to extend copyright protection to APIs means that third parties can discover and write APIs even for systems whose creator would prefer they remain proprietary. In some instances, standards that remain closed are reverse engineered and published by others, opening those standards to compatible programs.[263] That third-party publication sometimes leads the copyright holder to open an otherwise closed standard.[264] But even if it doesn't, third parties should be free to develop products that work with the standard. Cory Doctorow calls this "adversarial interoperability."[265] It is the critical piece of the interoperability puzzle that copyright rulings either foster or thwart.

Adversarial interoperability is the reason we have a vibrant and competitive software and hardware industry in the first place. Before the law endorsed interoperability, IBM was long the dominant maker of computer hardware, and it controlled what limited market for computer software existed in the early computer industry for the simple reason that only IBM knew the APIs necessary to make software run reliably on its computers. IBM controlled the market for PC-compatible computers in the early 1980s through its control of the IBM PC BIOS—the set of APIs that permitted the operating system to communicate with the computer's processor and other

261. HTML, CSS, and ECMAScript are examples.

262. *ECMAScript® 2017 Language Specification (ECMA-262, 8th edition, June 2017)*, ECMA INT'L, https://www.ecma-international.org/ecma-262/8.0/index.html [https://perma.cc/PMK5-683D].

263. *Apple Accessory Protocol*, NUXX, https://nuxx.net/wiki_archive/A/Apple_Accessory_Protocol [https://perma.cc/ZM3H-MHML] (disclosing reverse-engineered protocol for communicating with an iPod).

264. *See, e.g.*, *Office File Formats*, MICROSOFT (July 20, 2021), https://msdn.microsoft.com/en-us/library/office/cc313118(v=office.12).aspx [https://perma.cc/M35K-GHXC] (disclosing Microsoft Office file formats, which were previously proprietary); *Using the HomeKit Accessory Protocol Specification (Noncommercial Version)*, APPLE DEV., https://developers.apple.com/homekit/faq/ [https://perma.cc/7QH2-G2K3] (disclosing proprietary Apple HomeKit Accessory Protocol for noncommercial use).

265. *See* Cory Doctorow, *Adversarial Interoperability*, ELEC. FRONTIER FOUND. (Oct. 2, 2019), https://www.eff.org/deeplinks/2019/10/adversarial-interoperability [https://perma.cc/D43D-TVAN] (defining "adversarial interoperability" as "when you create a new product or service that plugs into the existing ones *without the permission* of the companies that make them").

hardware.[266] Software developed for the IBM PC was written to communicate using APIs provided by the IBM PC BIOS. In order to run that software, competing PC makers needed to provide their own BIOS that could use those APIs.

In 1984, a company called Phoenix Technologies reimplemented the IBM PC BIOS API in its own original software code, copying only the elements necessary for compatibility.[267] As with Java, those elements included a hierarchy of commands that were necessarily the same in both systems.[268] IBM did not take legal action against Phoenix, and the availability of the Phoenix BIOS led to a proliferation of IBM PC-compatible "clone" computers from Compaq, Dell, and others.[269] That in turn opened the door to multiple software companies to write programs for the now-accessible BIOS with the confidence that those programs would run on many different hardware systems, prompting the development of the modern independent software industry.

The law's openness to use of APIs has led, over time, to greater openness across the computer industry. Software for IBM computers was an IBM-specific business until Phoenix opened the BIOS, allowing IBM-compatible PCs. And because IBM had already published the DOS APIs to allow third parties to write programs for IBM PCs, those third parties could write for the compatible computers too.

Even after the IBM PC market opened to competition in the 1980s, the PC ecosystem was still not compatible with other operating system formats such as Apple or Linux. Once again, APIs led the way towards greater connectivity. For example, in 1993, open-source developer Andre Julliard released WINE, a program that allows Windows applications to run on computers that use the Linux operating system by translating calls to Windows APIs into the corresponding calls to Linux APIs.[270] To do so, it must use the same hierarchy of function names, just as Android does with respect to Java. Just as Linux users found it useful to be able to run some

266. The history of IBM in the early computer industry is well-documented. *See, e.g.*, CHARLES H. FERGUSON & CHARLES R. MORRIS, COMPUTER WARS: HOW THE WEST CAN WIN IN A POST-IBM WORLD 52–53 (1993) (describing IBM's early theories of protection against PC cloning); Russell Moy, *A Case Against Software Patents*, 17 SANTA CLARA COMPUT. & HIGH TECH. L.J. 67, 71 (2000) (tracing the history of the early computer industry and the role of IBM).

267. James Langdell, *Phoenix Says Its BIOS May Foil IBM's Lawsuits*, PC MAG., July 10, 1984, at 56, https://books.google.com/books?id=Bwng8NJ5fesC&lpg=PA6&pg=PA56#v=onepage&q&f=false [https://perma.cc/2XHM-XW7B].

268. INT'L BUS. MACH. CORP., IBM PERSONAL SYSTEM/2™ AND PERSONAL COMPUTER BIOS INTERFACE TECHNICAL REFERENCE 2-17 (1987), http://www.nj7p.org/Computers/IBM%20PC/work/BIOS_Interface_Technical_Reference.pdf [https://perma.cc/734X-3BBJ].

269. *Send in the Clones*, COMPUT. HIST. MUSEUM, http://www.computerhistory.org/revolution/personal-computers/17/302 [https://perma.cc/YJ6V-PS8S].

270. This example is drawn from Gratz & Lemley, *supra* note 171, at 611–12.

Windows programs they were familiar with, Windows users—particularly developers—found it useful to be able to run some Linux programs they were familiar with.

While WINE compatibility was one-way, it paved the way for compatibility in the opposite direction. In 2016, Microsoft released the Windows Subsystem for Linux, which allows Linux programs to run on Windows, translating API calls in real time to allow the programs to run unmodified. Microsoft engaged in a "clean room" implementation of the Linux kernel APIs to ensure that only the API structure, and not any of the implementing code, was copied.[271]

Apple remained a holdout, but there too APIs and the development of common internet standards have made it much more open than it was. In the 1990s, it was essentially impossible to move data or programs from Apple to PC or vice versa. While Apple still seeks to exercise more control over its systems than other computer makers, the development of open internet protocols (and, ironically, cross-platform languages like Java) broke down that insularity, opening the way for programs to work across the competing, historically-incompatible PC platforms. Once again, APIs led the way. The result has been something that seemed impossible only a couple of decades ago—the ability to move data and programs seamlessly across platforms.

Interoperability is at the backbone of the internet. Indeed, it is not an exaggeration to say that the internet is at its core a set of open APIs—a simple protocol that anyone can use to pass packets of data—any kind of data—back and forth across a computer network. The enormous success of the internet is attributable to the fact that any device and any program can connect to it just by following a simple protocol. That led to a greater diversity of programs and content than the world has ever seen before.[272]

Interoperability is also critical to the development of the Internet of Things ("IoT") that connects a wide array of devices beyond computers.[273]

271. *Windows Subsystem for Linux Overview*, MICROSOFT (Apr. 22, 2016), https://blogs.msdn.microsoft.com/wsl/2016/04/22/windows-subsystem-for-linux-overview/ [https://perma.cc/LDX9-4469] ("The drivers do not contain code from the Linux kernel but are instead a clean room implementation of Linux-compatible kernel interfaces.").

272. *See* Mark A. Lemley, *IP in a World Without Scarcity*, 90 N.Y.U. L. REV. 460, 470–71 (2015) (arguing the internet "fundamentally alter[ed] the economics of the creative industries"). In recent years the internet has been increasingly concentrated in the hands of a few platforms. Interoperability can help solve that problem too, as we discuss *infra*.

273. *See, e.g.*, Lu Tan & Neng Wang, *Future Internet: The Internet of Things*, 2010 3d Int'l Conf. Advanced Comput. Theory & Eng'g, Aug. 2010, at 379 ("Only if we can solve the interoperability problem [can we] have a . . . real Internet of Things."); *Developing the Interoperable Internet of Things*, OPEN CONNECTIVITY FOUND. (June 27, 2017), https://openconnectivity.org/blog/developing-interoperable-internet-things [https://perma.cc/5ZG9-G6TJ] (indicating that a common language is important to enable development for the interoperable IoT); Gary Eastwood, *IoT's Interoperability Challenge*, NETWORK WORLD (July 5,

IoT by definition depends on autonomous communication among a wide range of devices. That cannot happen without interoperable standards in the IoT market.[274] The importance of interoperability in the software and internet environments has been so clearly demonstrated, and so widely accepted, that most IoT programmers are writing only to open standards in the first place.[275] But even where parties contract for interoperability, for instance by using open-source software, legal interoperability plays a role. It allows downstream users to avoid an "anticommons" of overlapping and potentially conflicting contractual commitments.[276]

Interoperability, then, has been key to innovation in the software industry. The freedom to interoperate led directly to the unprecedented explosion of creativity in the software and internet industries in the 1990s and 2000s. Once programmers could write code that could be plugged into existing programs, they could compete to offer new tools without having to build an entire system from scratch or tying themselves to an existing walled garden.

Finally, interoperability will play an important role in establishing a right to repair. While consumers have long had the power to tinker with,

2017, 6:03 AM), https://www.networkworld.com/article/3205207/internet-of-things/iots-interoperability-challenge.html [https://perma.cc/U6YA-4HQQ] ("[I]nterconnectivity comes at a price, as the popularity increases and the number of devices and networks expands, the lack of interoperability between [devices] becomes an issue."); Giancarlo Fortino, Maria Ganzha, Carlos Palau & Marcin Paprzycki, *Interoperability in the Internet of Things*, COMPUTER.ORG (Dec. 2016), https://www.computer.org/publications/tech-news/computing-now/interoperability-in-the-internet-of-things [https://perma.cc/77BW-JX63] ("[M]any IoT researchers and industry leaders are focusing on interoperability."); *Interoperability: The Challenge Facing the Internet of Things*, PROPHET, https://www.prophet.com/thinking/2014/02/interoperability-the-challenge-facing-the-internet-of-things/ [https://perma.cc/GVG3-LBNN] ("[O]ne of the central-most challenges facing IoT is the enablement of seamless interoperability between each connection."); James Manyika, Michael Chui, Peter Bisson, Jonathan Woetzel, Richard Dobbs, Jacques Bughin & Dan Aharon, *Unlocking the Potential of the Internet of Things*, MCKINSEY DIGITAL (June 2015), http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world [https://perma.cc/3P4J-38AD] ("Interoperability between IoT systems is critical. Of the total potential economic value the IoT enables, interoperability is required for 40 percent on average and for nearly 60 percent in some settings."); Phillip Tracy, *IoT Interoperability: Where It Stands and What Comes Next*, RCRWIRELESS NEWS (Oct. 31, 2016), https://www.rcrwireless.com/20161031/internet-of-things/iot-interoperability-tag31-tag99 [https://perma.cc/2G5Z-2NJS] ("IoT ecosystems require interoperability to create seamless programmability of devices or sensors in enabling a world of connected devices.").

274. *See supra* note 273.

275. Brian Ray, *Open Source Software and Hardware for the Internet of Things*, IOT FOR ALL (June 8, 2017), https://medium.com/iotforall/open-source-software-and-hardware-for-the-internet-of-things-eca2aa728fa4 [https://perma.cc/W4G5-BG5C].

276. Clark D. Asay, *Software's Copyright Anticommons*, 66 EMORY L.J. 265 (2017); *see also* Clark D. Asay, *Copyright's Technological Interdependencies*, 18 STAN. TECH. L. REV. 189 (2015) (discussing how "conceptualizing copyright as an interdependent part of a creative system provides a more useful framework for analyzing the role of copyright, its interdependencies, and potential solutions related to creative processes").

modify, and repair the devices they own,[277] that freedom is under attack in the digital world, where restrictive terms of use purport to turn what seems to be a sale into a license of limited rights the copyright owner can take back.[278] But even if consumers have the legal right to repair their computer devices, jailbreak their iPhones, or mod their video games, their practical ability to do so is limited by access to modifications that will work with the original. Once again interoperability plays a key role, ensuring that people can write and release the add-ons, fixes, and spare parts needed to make a right to repair and tinker useful in practice.

The uncopyrightability of APIs isn't an accident or some sort of judicially created carve-out from the scope of copyrightable subject matter. As this section shows, it is part and parcel of the bargain copyright law strikes, protecting particular creative expression while taking care not to lock up ideas or functions. APIs serve an important function, and their availability has led to tremendous innovation—innovation that would have been stifled had copyright law been interpreted to give a software developer control over anything that might connect to or work with that software.

C. *The Complication of Partial Interoperability*

It is important to realize, however, that compatibility is always a matter of degree.[279] Compatibility is often only one-way, even when the same company makes the compatible products. This is often true of backwards compatibility, as when new platforms play old games, but new games can't be played on old platforms,[280] or when a new version of Microsoft Word can read documents written by the older version but not vice versa. Those are programs and devices issued by the same company, but they are often not perfectly compatible for the simple reason that if you add a new feature to a new system, an older system that lacks that feature can't implement it.

277. Pamela Samuelson, *Freedom to Tinker*, 17 THEORETICAL INQUIRIES L. 563, 564–65, 569 (2016).

278. *See generally* AARON PERZANOWSKI & JASON SCHULTZ, THE END OF OWNERSHIP: PERSONAL PROPERTY IN THE DIGITAL ECONOMY (2016) (asserting that licenses "mean that you don't own the thing you buy").

279. *See, e.g.*, BAND & KATOH, *supra* note 21, at 8 (giving the example of an application running on an operating system without a complete set of interface specifications that is "interoperable" but nonetheless not as interoperable as possible).

280. *See, e.g.*, Will White, Note, *Would You Like to Save Your Game?: Establishing A Legal Framework for Long-Term Digital Game Preservation*, 81 OHIO ST. L.J. 567, 580 n.91 (2020) (tying game obsolescence to upgrades in hardware and a lack of backwards compatibility); Rob Dolen, *PlayStation Could Beat Nintendo to the Punch with Backwards Compatibility*, GAMERANT (Apr. 1, 2021), https://gamerant.com/playstation-nintendo-backwards-compatibility-ps3-psp-ps-vita-playstation-now/ [https://perma.cc/3TFE-DN48] (comparing backwards compatibility among leading video game consoles, including Xbox, where it is a staple of the ecosystem, and PlayStation, which had scrapped backwards compatibility as a "much requested, little-used feature" but has moved to restore it).

Interoperability is more likely to be imperfect when it is implemented by third parties rather than the original developer of the interface. Sometimes developers of interfaces choose not to reveal all the details of its interfaces to those to whom it licenses the interfaces. Even if they intend to make the products fully compatible, it is harder to make two different programs work together seamlessly if they are written and implemented by different companies that can't easily coordinate on how their programs change. Perfect compatibility is less likely still in APIs that have been opened to the public.[281] Public interfaces may deliberately be limited by the company that publishes them.[282]

What Cory Doctorow calls "adversarial interoperability"[283] is even more commonly imperfect. In these cases, as in *Google*, one party is trying to write a program that connects to another party's system while the other party is at best not helping and at worst actively trying to sabotage the effort. In that case, generally the best we can hope for is one-way compatibility in which the new program emulates or draws data from the old. In the *Connectix* case, for instance, the defendant's software platform for playing Sony PlayStation games on a PC was only partly compatible with the PlayStation.[284] And in *Lotus v. Borland*, Borland's Quattro Pro spreadsheet allowed users to copy their pre-written macros to use in the new system but didn't fully emulate Lotus 1-2-3.[285]

In the *Google* case, significant differences existed in the types of tasks needed for the computing platforms for which the Java API was initially developed (e.g., enabling a mouse) as compared with the Android smartphone platform (e.g., enabling GPS). Google and Android app developers therefore did not need or want total compatibility with programs written for laptops with the full 166 Java API packages. And Google also wanted to develop a new and better system that served a variety of purposes beyond just implementing Java code. In that circumstance, as in many others, there may be no realistic option but to "fork" the code with partial or one-way compatibility.

281. BAND & KATOH, *supra* note 21, at 8.

282. *See* Complaint at 21, FTC v. Facebook, Inc., No. 1:20-cv-03590 (D.D.C. June 28, 2021), 2021 WL 2643627 (alleging that Facebook's imposition and enforcement of conditions on access to APIs "in order to suppress and deter competitive threats to its personal social networking monopoly" was an illegal anticompetitive practice). Facebook's motion to dismiss was granted in June. FTC v. Facebook, Inc., No. 1:20-cv-03590 (D.D.C. June 28, 2021), 2021 WL 2643627.

283. Doctorow, *supra* note 265.

284. *See* Sony Comput. Ent., Inc. v. Connectix Corp., 203 F.3d 596, 599 (9th Cir. 2000) (stating that defendant's software "emulates" the functioning of the PlayStation console, but it does not play PlayStation games as well).

285. Lotus Dev. Corp v. Borland Int'l, Inc., 49 F.3d 807, 810 (1st Cir. 1995).

The fact that compatibility is incomplete does not mean, as the Federal Circuit wrongly suggested,[286] that it isn't important. One-way compatibility allows companies to improve on software programs without customers being locked into an obsolete system because of the investment they have made in learning it or storing their data there.[287] It establishes a base of compatibility that can make interaction between systems easier even if they aren't fully compatible (as anyone who has copied and pasted text across programs using standard tools can attest). It enables new code to be more readily adopted by allowing developers to draw on their existing skills and knowledge in writing code for the new platform. And as we discuss in the next section, it makes it harder for a dominant firm to prevent competition or extend its control into adjacent markets.

Partial compatibility will also become increasingly important to historians and archivists. Software platforms change rapidly, and more and more content and code is written in obsolete languages and stored on obsolete hardware. Interoperability permits archivists and others to rescue and restore obsolete programs, games, and text and to make sure that old content can remain accessible and relevant on new platforms.[288]

D. Beyond Copyright: Contract, Antitrust, and Interoperability

As we noted in the last two parts, interoperability has substantial benefits for consumers and innovation. But arguably its most important benefit concerns market structure and competition. Interoperability benefits smaller competitors and newer companies by preventing established incumbents from tying up existing customers and so locking them out of the market.

Interoperability is particularly important to startups. Companies that develop apps for mobile phones, for instance, are often small. They may not have the ability to write several different versions of a program from scratch, one for each hardware platform or incompatible programming language, much less to separately negotiate agreements with each such platform provider in the economy. By allowing an app developer to reach the widest possible market, legal protection for interoperability increases the number of

286. *Oracle V*, 886 F.3d 1179, 1206 (Fed. Cir. 2018).

287. On problems of lock-in effects, see Mark A. Lemley & David McGowan, *Legal Implications of Network Economic Effects*, 86 CALIF. L. REV. 479 (1998); CARL SHAPIRO & HAL R. VARIAN, INFORMATION RULES: A STRATEGIC GUIDE TO THE NETWORK ECONOMY (1999); Joseph Farrell & Paul Klemperer, *Coordination and Lock-In: Competition with Switching Costs and Network Effects*, *in* 3 HANDBOOK OF INDUSTRIAL ORGANIZATION 1967 (Mark Armstrong & Robert Porter eds., 2007); S.J. Liebowitz & Stephen E. Margolis, *The Fable of the Keys*, 33 J.L. & ECON. 1 (1990).

288. For a discussion of this problem, see Mark A. Lemley, *Disappearing Content*, 101 B.U. L. REV. 1255 (2021).

creative new works produced each year. It also ensures that no one company, no matter how dominant its platform, gets to decide what web pages you can access, what files you can share, or what programs you can download.

Without the legal ability to use APIs, software developers would be at the mercy of platform and programming giants who could decide whether, when, and how anyone could write or use a computer program that ran on their system. Startups will not invest in new products—for mobile phones or video games or the Internet of Things—without confidence that their products will work on the dominant platforms. That is why the risk of overprotecting copyright is so much greater in software than in other areas. Giving too much protection to a song may incrementally discourage the creation of somewhat similar songs. Giving copyright owners control over interoperability risks shutting down the software development ecosystem altogether.

Conversely, interoperability offers the potential of opening the "walled gardens" that have come to dominate much of the internet today.[289] Facebook, Apple, Microsoft, and (to a lesser extent) Amazon and Google all benefit from a large customer base locked into their systems by investment, choice, learning, or simply inertia. For a variety of reasons, the wave of Schumpeterian competition that has periodically dethroned prior dominant firms (IBM, AT&T, Yahoo!, MySpace et al.) has stalled in the last two decades.[290] But interoperability offers the promise of restarting that competition, allowing, for instance, a new, privacy-focused social media company to grow its own customer base while connecting into the Facebook and Instagram networks of those customers.[291] The new European Digital

289. *See, e.g.*, Hunter, *supra* note 18 (explaining that "legal scholarship is locked up inside the 'walled gardens' of commercial databases"); Mehra, *supra* note 18 (asserting that "[i]n the worlds of technology and cyberlaw, the term 'walled garden' has become an epithet to epitomize a proprietary (and likely sterile) community"); Lastowka, *supra* note 18 (discussing digital walled gardens in the context of copyright law).

290. For one explanation, and some suggestions about what can be done about it, see Mark A. Lemley & Andrew McCreary, *Exit Strategy*, 101 B.U. L. REV. 1 (2021).

291. *See, e.g.*, Michael Kades & Fiona M. Scott Morton, Interoperability as a Competition Remedy for Digital Networks (February 2021) (unpublished manuscript) (on file with authors), https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3808372 [https://perma.cc/W3S6-HWQL] (explaining how an interoperability requirement can remedy monopolization by engendering robust, disruptive marketplace competition); Cory Doctorow, *Tech Trustbusting's Moment Has Arrived*, PLURALISTIC, PLURALISTIC (Feb. 20, 2021), https://pluralistic.net/2021/02/20/escape-velocity/#trustbusting-time [https://perma.cc/9RYJ-4X5G] (suggesting means of government and community action to protect interoperators for the sake of competition); Bennett Cyphers & Cory Doctorow, *Privacy Without Monopoly: Data Protection and Interoperability*, ELEC. FRONTIER FOUND. (June 11, 2021), https://www.eff.org/files/2021/06/14/privacy_without_monopoly.pdf [https://perma.cc/HYK5-KKHG] (unpacking how competitive pressures derived from interoperability could foster better consumer control over privacy data).

Markets Act requires operating systems to permit interoperability and requires device makers to allow sideloading of apps.[292]

Opening up the walled gardens will take more than just getting copyright law right. Companies that want to close their systems have used standard-form contracts, terms of service, and even a federal computer crime statute to try to prevent competitors from writing interoperable programs. Protecting interoperability means turning away contract and Computer Fraud and Abuse Act (CFAA) claims that dominant sites have used to prevent third parties from offering products or services that interconnect with dominant firm sites.[293] It may also mean favoring antitrust enforcement that demands structural separation, or at least imposes nondiscrimination rules on self-dealing by vertically integrated monopolists,[294] and perhaps using antitrust or

292. *Proposal for a Regulation of the European Parliament and of the Council on Contestable and Fair Markets in the Digital Sector (Digital Markets Act)*, at arts. 5(e), 6(1)(c), COM (2020) 842 final (Dec. 15, 2020), https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:52020PC0842&from=en [https://perma.cc/5J9M-VSDS].

293. *See, e.g.*, Facebook, Inc. v. Power Ventures, Inc., 844 F.3d 1058 (9th Cir. 2016) (holding that Facebook could use the CFAA to prevent Power Ventures from scraping its social graph); Jonathan Mayer, *Cybercrime Litigation*, 164 U. PA. L. REV. 1453 (2016) (arguing that computer abuse legislation, including the CFAA, is overbroad and ineffective); Orin S. Kerr, *Cybercrime's Scope: Interpreting "Access" and "Authorization" in Computer Misuse Statutes*, 78 N.Y.U. L. REV. 1596 (2003) (discussing why courts have construed unauthorized access statutes in an overly broad manner that threatens to criminalize a surprising range of innocuous conduct involving computers). More recently, courts in *HiQ v. LinkedIn* and *Sandvig v. Barr* have declined to apply the CFAA to crawling of a public web page in violation of terms of service. HiQ Labs, Inc. v. LinkedIn Corp., 938 F.3d 985 (9th Cir. 2019); Sandvig v. Barr, 451 F. Supp. 3d 73 (D.D.C. 2020). The Supreme Court decision this term in *Van Buren v. United States*, No. 19-783 (U.S. June 3, 2021), limiting the reach of the CFAA is a big step forward for interoperability. Without terms of service and CFAA laws standing in the way, social media companies could help users port their friends and history to a new platform, making competing platforms more realistic. *See, e.g.*, Charles Duan, *Hacking Antitrust: Competition Policy and the Computer Fraud and Abuse Act*, 19 COLO. TECH. L.J. (forthcoming 2021) (explaining that the broad reading of the CFAA enables companies, social media platforms in particular, to stop competitors from building competing services); Mike Masnick, *Protocols, Not Platforms: A Technological Approach to Free Speech*, KNIGHT FIRST AMEND. INST. COLUM. U. (Aug. 21, 2019), https://knightcolumbia.org/content/protocols-not-platforms-a-technological-approach-to-free-speech [https://perma.cc/LD6G-YMEF] ("Moving us back toward a world where protocols are dominant over platforms could be of tremendous benefit to free speech and innovation online."); Corynne McSherry, *Want More Competition in Tech? Get Rid of Outdated Computer, Copyright, and Contract Rules*, ELEC. FRONTIER FOUND. (Dec. 20, 2018), https://www.eff.org/deeplinks/2018/12/want-more-competition-tech-get-rid-outdated-computer-copyright-and-contract-rules [https://perma.cc/7JXC-5JC9] (explaining how the CFAA harms competition by allowing companies to legally block cross-platform accessibility tools); Thomas E. Kadri, *Platforms as Blackacres*, 68 UCLA L. Rev. (forthcoming Feb. 2021) ("[P]latforms have used cyber-trespass law to . . . block[] or chill[] academics, journalists, and competitors from accessing information they need to research digital life.").

294. Khan, *supra* note 18. The problem of discriminatory refusals to deal by vertically integrated monopolists is particularly stark when the monopolist excludes access to a competitor. Apple, for instance, has made it difficult for competing music services like Spotify to run on its phones, and has banned "side-loading" altogether, requiring that all apps on the phone be sold

other tools to force interoperability.[295] But none of those efforts will do much good at encouraging compatibility and breaking the hold dominant tech incumbents have on their markets if copyright stands in the way. Copyright law standing alone can't ensure interoperability. But a misinterpretation of copyright law could doom it.

It may seem ironic that a Supreme Court victory for *Google* holds the key to undercutting the power of modern tech firms. But interoperability has always worked to benefit upstarts and new entrants, whether it was IBM clones thirty years ago, Android a dozen years ago, or those who would like to bypass smartphone app stores today.

Conclusion

The Supreme Court in *Google v. Oracle* took an important step in protecting the vibrant, open software development process from the threat of a resurgent copyright law. But it also missed an important opportunity to make clear what most courts had settled twenty-five years ago but which is once again being debated: the functional elements of APIs aren't copyrightable. Courts should refuse to allow anyone to control the components necessary to make programs work together.

through the app store, taking a whopping thirty percent of all their revenue, and refusing to allow apps that might circumvent that monopoly. Complaint for Injunctive Relief at para 3., Epic Games, Inc. v. Apple Inc., 439 F. Supp. 3d 817 (N.D. Cal. 2020) (Case No. 4:20-cv-05640-YGR); Seth Schiesel, *Apple Rejects Facebook's Gaming App, for at Least the Fifth Time*, N.Y. TIMES (June 18, 2020), https://www.nytimes.com/2020/06/18/technology/apple-ios-facebook-gaming-app.html [https://perma.cc/JS9E-PBAH]; Damien Geradin & Dimitrios Katsifis, *The Antitrust Case Against the Apple App Store*, J. COMPETITION L. & ECON. (forthcoming 2021). Recent House bills require platforms not to discriminate in favor of their own apps. *See* American Choice and Innovation Online Act, H.R. 3816, 117th Cong. (2021) (providing that certain discriminatory conduct by covered platforms should be unlawful); Ending Platform Monopolies Act, H.R. 3825, 117th Cong. (2021) (promoting competition and economic opportunity in digital markets by eliminating the conflicts of interest that arise from dominant online platforms' concurrent ownership or control of an online platform and certain other businesses); ACCESS Act of 2021, H.R. 3849, 117th Cong. (2021) (requiring a covered platform to maintain a set of transparent, third-party-accessible interfaces to facilitate and maintain interoperability with a competing business).

295. *See* Chinmayi Sharma, *Concentrated Digital Markets, Restrictive APIs, and the Fight for Internet Interoperability*, 50 U. MEM. L. REV. 441 (2019) (discussing how formal antitrust investigations by the FTC could help shape acceptable standards for API design and expectations for online interoperability); Cyphers & Doctorow, *supra* note 291 (proposing a policy to force interoperability).